FILED
2013 Jun-12 PM 02:40
U.S. DISTRICT COURT
N.D. OF ALABAMA

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF ALABAMA

Intellectual Ventures II LLC,	**)**	
a limited liability company,	**)**	
	)	
Plaintiff,	**)**	
v.	**)**	**Case No. ___________________**
	)	
BBVA Compass Bancshares, Inc.,	**)**	**(Demand for Jury Trial)**
and Compass Bank N.A. d/b/a	**)**	
BBVA Compass,	**)**	
	)	
Defendants.	**)**	

COMPLAINT FOR PATENT INFRINGEMENT

COMES NOW Plaintiff Intellectual Ventures II LLC and for its complaint against Defendants BBVA Compass Bancshares, Inc. and Compass Bank N.A., doing business as BBVA Compass, hereby alleges as follows:

THE PARTIES

1. Plaintiff Intellectual Ventures II LLC ("Intellectual Ventures II") is a Delaware limited liability company having its principal place of business located at 3150 139th Avenue SE, Bellevue, Washington, 98005.

2. Upon information and belief, BBVA Compass Bancshares, Inc. ("BBVA Compass Bancshares") is a financial holding company with its principal place of business at 15 South 20th Street, Birmingham, Alabama 35233.

3. Upon information and belief, BBVA Compass Bancshares has its headquarters in this judicial district and transacts substantial business within this judicial district.

4. Upon information and belief, Compass Bank, N.A., doing business as BBVA Compass ("BBVA Compass Bank"), is a national banking association with its principal place of business at 15 South 20th Street, Birmingham, Alabama 35233. Upon information and belief, BBVA Compass Bank is a wholly owned subsidiary of BBVA Compass Bancshares. BBVA Compass Bancshares and BBVA Compass Bank will be referred to herein collectively as "BBVA Compass."

5. Upon information and belief, BBVA Compass Bank has its headquarters in this judicial district and transacts substantial business within this judicial district.

6. BBVA Compass offers banking services to individuals and business in the United States, including Alabama, and particularly within this district. BBVA Compass provides online banking services via electronic means including, but not limited to, the web site https://www.bbvacompass.com. In connection with these online banking services and other systems and services, BBVA Compass infringes one or more claims of United States Patent No. 5,745,574 ("the '574 Patent"); United States Patent No. 6,826,694 ("the '694 Patent""); United States Patent No. 6,715,084 ("the '084 Patent); United States Patent No. 6,314,409

("the '409 Patent"); and United States Patent No. 7,634,666 ("the '666 Patent") (collectively the "Patents-in-Suit").

JURISDICTION AND VENUE

7. This is a civil action for patent infringement under the Patent Laws of the United States, 35 U.S.C. § 1 *et. seq*. This Court has subject matter jurisdiction under 28 U.S.C. §§1331 and 1138(a).

8. This Court has general personal jurisdiction over BBVA Compass because it has its headquarters and does substantial and continuous business in this judicial district. This Court has specific jurisdiction over BBVA Compass because it has committed acts giving rise to this action and has established minimum contacts within this judicial district such that the exercise of jurisdiction over BBVA Compass would not offend traditional notions of fair play and substantial justice.

9. Venue is proper in this judicial district pursuant to 28 U.S.C. §§1391(b)-(c) and 1400(b) because BBVA Compass has conducted business in this district and/or provided services to its customers within this judicial district, and has committed acts of patent infringement within this district giving rise to this action.

INTELLECTUAL VENTURES AND THE PATENTS-IN-SUIT

10. Intellectual Ventures Management, LLC ("Intellectual Ventures") was founded in 2000. Since its founding, Intellectual Ventures has been deeply involved in the business of invention. Intellectual Ventures creates inventions and files patent applications for those inventions; collaborates with others to develop and patent inventions; and acquires and licenses patents from individual inventors, universities and other institutions. A significant aspect of Intellectual Ventures' business is managing the Plaintiff in this case, Intellectual Ventures II.

11. Intellectual Ventures' business includes purchasing important inventions from individual inventors and institutions and then licensing the inventions to those who need them. Through this business, Intellectual Ventures allows inventors to reap a financial reward from their innovations, which is frequently difficult for individual inventors to do. To date, Intellectual Ventures has acquired more than 70,000 IP assets and, in the process, has paid individual inventors hundreds of millions of dollars for their inventions. Intellectual Ventures, in turn, has earned more than $3 billion by licensing these patents to some of the world's most innovative and successful technology companies who continue to use them to make computer equipment, software, semiconductor devices, and a host of other products.

12. Intellectual Ventures also creates inventions. Intellectual Ventures has a staff of scientists and engineers who develop ideas in a broad range of fields, including agriculture, computer hardware, life sciences, medical devices, semiconductors, and software. Intellectual Ventures has invested millions of dollars developing such ideas and has filed hundreds of patent applications on its inventions every year, making it one of the top patent filers in the world. Intellectual Ventures has also invested in laboratory facilities to assist with the development and testing of new ideas.

13. Intellectual Ventures also creates inventions by collaborating with inventors and research institutions around the world. For example, Intellectual Ventures has developed inventions by selecting a technical challenge, requesting proposals for inventions to solve the challenge from inventors and institutions, selecting the most promising ideas, rewarding the inventors and institutions for their contributions, and filing patent applications on the ideas. Intellectual Ventures has invested millions of dollars in this way and has created a network of more than 4,000 inventors worldwide.

14. On April 28, 1998, the ’574 Patent, titled “Security Infrastructure For Electronic Transactions,” was duly and lawfully issued by the United States patent and Trademark Office (“PTO”). A copy of the ’574 Patent is attached hereto as Exhibit A.

15. On November 30, 2004, the ’694 Patent, titled “High Resolution Access Control,” was duly and lawfully issued by the PTO. A copy of the ’694 Patent is attached hereto as Exhibit B.

16. On March 30, 2004, the ’084 Patent, titled “Firewall System And Method Via Feedback From Broad-Scope Monitoring For Intrusion Detection,” was duly and lawfully issued by the PTO. A copy of the ’084 Patent is attached hereto as Exhibit C.

17. On November 6, 2001, the ’409 Patent, titled “System For Controlling Access And Distribution of Digital Property,” was duly and lawfully issued by the PTO. A copy of the ’409 Patent is attached hereto as Exhibit D.

18. On December 15, 2009, the ’666 Patent, titled “Crypto-Engine For Cryptographic Processing Of Data,” was duly and lawfully issued by the PTO. A copy of the ’666 Patent is attached hereto as Exhibit E.

19. Intellectual Ventures II is the owner and assignee of all right, title and interest in and to the Patents-in-Suit and holds the right to sue and recover damages for infringement thereof, including past damages.

Count I

<u>INFRINGEMENT OF U.S. PATENT NO. 5,745,574</u>

20. Paragraphs 1-19 are reincorporated by reference as if fully set forth herein.

21. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass has directly infringed and continues to directly infringe, literally and/or under the doctrine of equivalents, at least claim 23 of the ’574 Patent by making, using, providing, systems and services that comply with the PCI Data Security Standard for encrypting data during communication sessions, including, but not limited to, its website.

22. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass also has and continues to indirectly infringe at least claim 23 of the ’574 Patent by inducing others to infringe and/or contributing to the infringement of others, including third party users of such systems and services in this judicial district and elsewhere in the United States. Specifically, Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass has actively induced and continues to induce the infringement of at least claim 23 of the ’574 Patent at least by actively inducing the use of such systems and services by third party users in the United States. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass knew or should have known that its conduct would induce others to encrypt data during a communication sessions in a manner that infringes the ’574 Patent. Intellectual Ventures II is informed and believes, and thereon alleges, that these third parties have infringed and will continue to infringe the ’574 Patent in violation of 35 U.S.C. § 271(a) by

using the infringing system. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass through at least its website at https://www.bbvacompass.com actively induced its customers to infringe the ’574 Patent.

23. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass has contributorily infringed and continues to contributorily infringe at least claim 23 of the ’574 Patent by providing within the United States infringing systems and services that constitute a material part of the claimed invention and are not staple articles of commerce suitable for substantial non-infringing use. Intellectual Ventures II is informed and believes, and thereon alleges, that these third parties have infringed and will infringe the ’574 Patent in violation of 35 U.S.C. § 271(a) by using the infringing system.

24. Intellectual Ventures II has provided written notice via a letter to BBVA Compass of its infringement of at least claim 23, and BBVA Compass also has written notice of its infringement by virtue of the filing and service of this Complaint.

25. Intellectual Ventures II has suffered damages as a result of BBVA Compass’ infringement of the ’574 Patent in an amount to be proven at trial.

Count II

INFRINGEMENT OF U.S. PATENT NO. 6,826,694

26. Paragraphs 1-19 are reincorporated by reference as if fully set forth herein.

27. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass has directly infringed and continues to directly infringe, literally and/or under the doctrine of equivalents, at least claim 1 of the ’694 Patent by making, using, providing, offering to sell and/or selling its Small Business Merchant system/service.

28. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass also has and continues to indirectly infringe at least claim 1 of the ’694 Patent by inducing others to infringe and/or contributing to the infringement of others, including third party users of its Small Business Merchant service/system in this judicial district and elsewhere in the United States. Specifically, Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass has actively induced and continues to induce the infringement of at least claim 1 of the ’694 Patent at least by actively inducing the use of its Small Business Merchant system/service by third party users in the United States. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass knew or should have known that its conduct would

induce others to use these systems in a manner that infringes the ’694 Patent. Intellectual Ventures II is informed and believes, and thereon alleges, that these third parties have infringed and will infringe the ’694 Patent in violation of 35 U.S.C. § 271(a) by using the infringing system. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass through at least its website at https://bbvacompass.com actively induced its customers to infringe the ’694 Patent.

29. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass has contributorily infringed and continues to contributorily infringe at least claim 1 of the ’694 Patent by providing, selling and/or offering to sell within the United States infringing products that constitute a material part of the claimed invention and are not staple articles of commerce suitable for substantial non-infringing use. Intellectual Ventures II is informed and believes, and thereon alleges, that these third parties have infringed and will infringe the ’694 Patent in violation of 35 U.S.C. § 271(a) by using the infringing system.

30. Intellectual Ventures II has provided written notice via a letter to BBVA Compass of its infringement of at least claim 1, and BBVA Compass also has written notice of its infringement by virtue of the filing and service of this Complaint.

31. Intellectual Ventures II has suffered damages as a result of BBVA Compass' infringement of the '694 Patent in an amount to be proven at trial.

Count III

<u>INFRINGEMENT OF U.S. PATENT NO. 6,715,084</u>

32. Paragraphs 1-19 are reincorporated by reference as if fully set forth herein.

33. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass has directly infringed and continues to directly infringe, literally and/or under the doctrine of equivalents, at least claim 26 of the '084 Patent by making, using, offering to sell and/or selling its Small Business Merchant system/service that uses PCI Data Security Standard compliant intrusion detection and prevention.

34. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA also has and continues to indirectly infringe at least claim 26 of the '084 Patent by inducing others to infringe and/or contributing to the infringement of others, including third party users of its Small Business Merchant system/service in this judicial district and elsewhere in the United States. Specifically, Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass has actively induced and continues to induce the infringement of at least claim 26 of the '084 Patent at least by actively inducing the

use of its Small Business Merchant system/service by third party users in the United States. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass knew or should have known that its conduct would induce others to use its Small Business Merchant system/service in a manner that infringes the ’084 Patent. Intellectual Ventures II is informed and believes, and thereon alleges, that these third parties have infringed and will infringe the ’084 Patent in violation of 35 U.S.C. § 271(a) by using the infringing system. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass through at least its website https://www.bbvacompass.com actively induced its customers to infringe the ’084 Patent.

35. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass has contributorily infringed and continues to contributorily infringe at least claim 26 of the ’084 Patent by providing, selling and/or offering to sell within the United States infringing systems and services that constitute a material part of the claimed invention and are not staple articles of commerce suitable for substantial non-infringing use. Intellectual Ventures II is informed and believes, and thereon alleges, that these third parties have infringed and will infringe the ’084 Patent in violation of 35 U.S.C. § 271(a) by using the infringing system.

36. Intellectual Ventures II has provided written notice via a letter to BBVA Compass of its infringement of at least claim 26, and BBVA Compass also has written notice of its infringement by virtue of the filing and service of this Complaint.

37. Intellectual Ventures II has suffered damages as a result of BBVA Compass' infringement of the '084 Patent in an amount to be proven at trial.

Count IV

<u>INFRINGEMENT OF U.S. PATENT NO. 6,314,409</u>

38. Paragraphs 1-19 are reincorporated by reference as if fully set forth herein.

39. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass has directly infringed and continues to directly infringe, literally and/or under the doctrine of equivalents, at least claim 38 of the '409 Patent by making, using, providing, offering to sell and/or selling its Mobility Pack and services that use PCI Data Security Standard technology for protecting customer information and account data.

40. Intellectual Ventures II is informed and believes, and thereon alleges, that First National also has and continues to indirectly infringe at least claim 38 of the '409 Patent by inducing others to infringe and/or contributing to the infringement of others, including third party users of such systems and services in

this judicial district and elsewhere in the United States. Specifically, Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass has actively induced and continues to induce the infringement of at least claim 38 of the ’409 Patent at least by actively inducing the use of such systems and services in the United States. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass knew or should have known that its conduct would induce others to use its systems and services in a manner that infringes the ’409 Patent. Intellectual Ventures II is informed and believes, and thereon alleges, that these third parties have infringed and will infringe the ’409 Patent in violation of 35 U.S.C. § 271(a) by using the infringing system. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass through at least its website https://www.bbvacompass.com actively induced its customers to infringe the ’409 Patent.

41. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass has contributorily infringed and continues to contributorily infringe at least claim 38 of the ’409 Patent by providing, selling and/or offering to sell within the United States infringing systems and services that constitute a material part of the claimed invention and are not staple articles of commerce suitable for substantial non-infringing use. Intellectual Ventures II is informed and believes, and thereon alleges, that these third parties have infringed and will

infringe the ’409 Patent in violation of 35 U.S.C. § 271(a) by using the infringing system.

42. Intellectual Ventures II has provided written notice via a letter to BBVA Compass of its infringement of at least claim 38, and BBVA Compass also has written notice of its infringement by virtue of the filing and service of this Complaint.

43. Intellectual Ventures II has suffered damages as a result of BBVA Compass’ infringement of the ’409 Patent in an amount to be proven at trial.

Count V

INFRINGEMENT OF U.S. PATENT NO. 7,634,666

44. Paragraphs 1-19 are reincorporated by reference as if fully set forth herein.

45. Intellectual Ventures II is informed and believes, and thereon alleges, that BBVA Compass has directly infringed and continues to directly infringe, literally and/or under the doctrine of equivalents, at least claim 4 of the ’666 Patent by using systems or services that use IBM System z mainframes.

46. Intellectual Ventures II has provided written notice via a letter to BBVA Compass of its infringement of at least claim 4, and BBVA Compass also has written notice of its infringement by virtue of the filing and service of this Complaint.

47. Intellectual Ventures II has suffered damages as a result of BBVA Compass' infringement of the '666 Patent in an amount to be proven at trial.

PRAYER FOR RELIEF

WHEREFORE, Intellectual Ventures II respectfully prays that this Court:

A. Enter judgment in favor of Intellectual Ventures II that BBVA Compass has infringed the '574 Patent;

B. Enter judgment in favor of Intellectual Ventures II that BBVA Compass has infringed the '694 Patent;

C. Enter judgment in favor of Intellectual Ventures II that BBVA Compass has infringed the '084 Patent;

D. Enter judgment in favor of Intellectual Ventures II that BBVA Compass has infringed the '409 Patent;

E. Enter judgment in favor of Intellectual Ventures II that BBVA Compass has infringed the '666 Patent

F. Enter judgment that Intellectual Ventures II be awarded damages adequate to compensate it for BBVA Compass' past infringement and any continuing or future infringement of the Patents-in-Suit up until the date such judgment is entered, including pre-judgment and post-judgment interest, costs and disbursements as justified under 35

U.S.C. § 284 and, if necessary, to adequately compensate Intellectual Ventures II for BBVA Compass' infringement, an accounting;

G. Enter judgment that Intellectual Ventures II be awarded attorney fees, costs and expenses incurred in prosecuting this action; and

H. Order that Intellectual Ventures II be granted such other, different, and additional relief as this Court deems equitable and proper under the circumstances.

DEMAND FOR JURY TRIAL

Plaintiff Intellectual Ventures II hereby demands trial by jury as to all issues so triable in this civil action.

Dated: June 12, 2013.

Respectfully submitted,

/s/ A.H. "Nick" Gaede, Jr.
A.H. "Nick" Gaede, Jr. (ASB-9661-G64A)

/s/ Jennifer A. Hanson
Jennifer A. Hanson (ASB-2100-E58H)

BAINBRIDGE, MIMS, ROGERS & SMITH, LLP
The Luckie Building, Suite 415
600 Luckie Drive (35223)
Post Office Box 530886
Birmingham, Alabama 35253
Telephone: (205) 879-1100
Facsimile: (205) 879-4300
ngaede@bainbridgemims.com
jhanson@bainbridgemims.com

OF COUNSEL:

Ian Feinberg
Elizabeth Day
Marc Belloli
FEINBERG DAY ALBERTI & THOMPSON LLP
1600 El Camino Real, Suite 280
Menlo Park, CA 94025
Direct: 650-618-4360
Fax: 650-618-4368
ifeinberg@feinday.com
eday@feinday.com
mbelloli@feinday.com
(*pro hac vice* applications forthcoming)

Counsel for Plaintiff

<u>DEFENDANTS TO BE PERSONALLY SERVED AT THE BELOW:</u>

BBVA Compass Bancshares, Inc.
B.S. Clanton
15 South 20th Street
Birmingham, Alabama 35296

Compass Bank, N.A. d/b/a BBVA Compass
15 South 20th Street
Birmingham, Alabama 35233

EXHIBIT A

US005745574A

United States Patent [19]

Muftic

[11] **Patent Number: 5,745,574**

[45] **Date of Patent: Apr. 28, 1998**

[54] **SECURITY INFRASTRUCTURE FOR ELECTRONIC TRANSACTIONS**

[75] Inventor: **Sead Muftic**, Hasselby, Sweden

[73] Assignee: **Entegrity Solutions Corporation**, San Jose, Calif.

[21] Appl. No.: **573,025**

[22] Filed: **Dec. 15, 1995**

[51] **Int. Cl.**^{6} **H04K 1/00**

[52] **U.S. Cl.** **380/23**; 380/25

[58] **Field of Search** 380/23, 24, 25, 380/21, 49

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,200,770	4/1980	Hellman et al. .	
4,218,582	8/1980	Hellman et al. .	
4,405,829	9/1983	Rivest et al. .	
4,424,414	1/1984	Hellman et al. .	
4,562,305	12/1985	Gaffney, Jr. .	
4,995,082	2/1991	Schnorr .	
5,214,702	5/1993	Fischer	380/23
5,347,580	9/1994	Molva et al.	380/25
5,369,705	11/1994	Bird et al.	380/21
5,420,927	5/1995	Micali	380/23
5,497,421	3/1996	Kaufman et al.	380/23
5,509,071	4/1996	Petrie et al.	380/21
5,604,804	2/1997	Micali	380/23
5,606,617	2/1997	Brands	380/30

OTHER PUBLICATIONS

Philip Zimmerman, "Pretty Good(tm) Privacy Public Key Encryption for the Masses", RGP Version 2.6.2 —Oct. 11, 1994.

"Privacy Enhancement for Internet Electronic Mail", Network Working Group, Request for Comments: 1421.

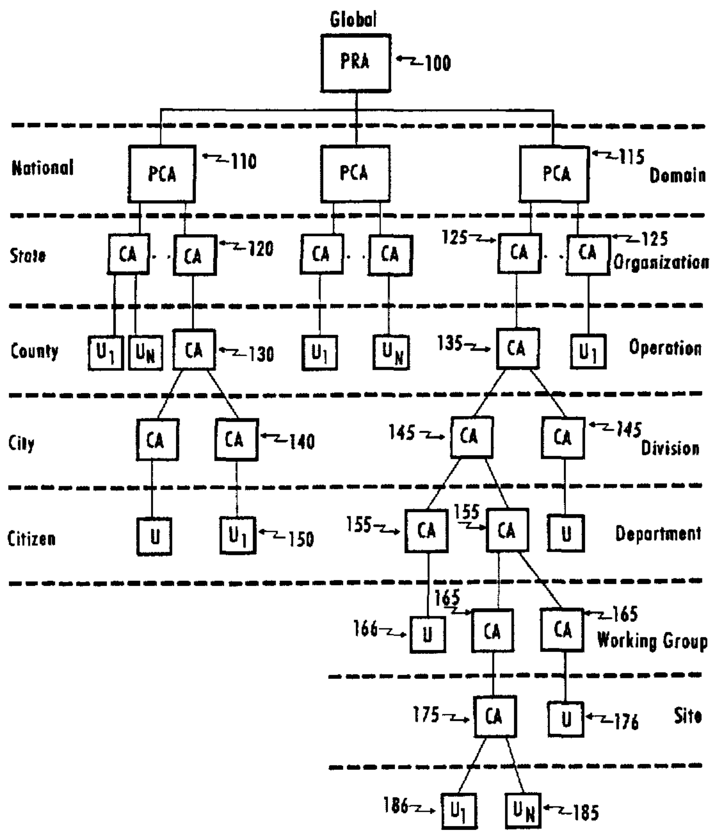
"Privacy Enhancement for Internet Electronic Mail", Network Working Group, Request for Comments: 1422.

Primary Examiner—David C. Cain
Attorney, Agent, or Firm—Lowe, Price, LeBlanc & Becker

[57] **ABSTRACT**

A plurality of certification authorities connected by an open network are interrelated through an authentication and certification system for providing and managing public key certificates. The certification system with its multiple certification and its policies constitute a public key infrastructure facilitating secure and authentic transactions over an unsecure network. Security services for applications and users in the network are facilitated by a set of common certification functions accessible by well-defined application programming interface which allows applications to be developed independently of the type of underlying hardware platforms used, communication networks and protocols and security technologies.

34 Claims, 29 Drawing Sheets



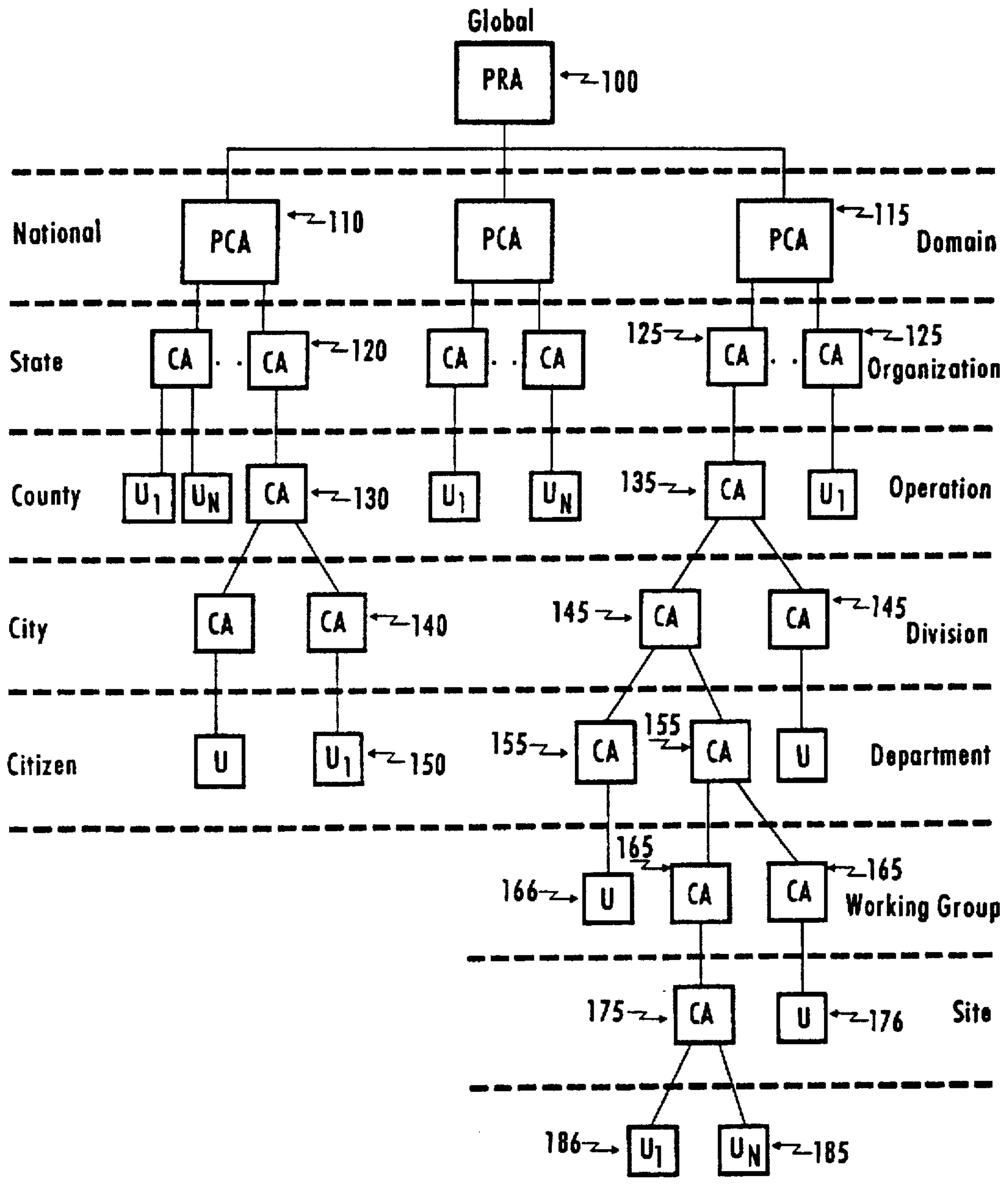

Global
PRA
100
National
PCA
110
PCA
PCA
115
Domain
State
CA
CA
120
CA
CA
125
CA
CA
125
Organization
County
U1
UN
CA
130
U1
UN
135
CA
U1
Operation
City
CA
CA
140
145
CA
CA
145
Division
Citizen
U
U1
150
155
CA
155
CA
U
Department
166
U
165
CA
CA
165
Working Group
175
CA
U
176
Site
186
U1
UN
185

Figure 1A

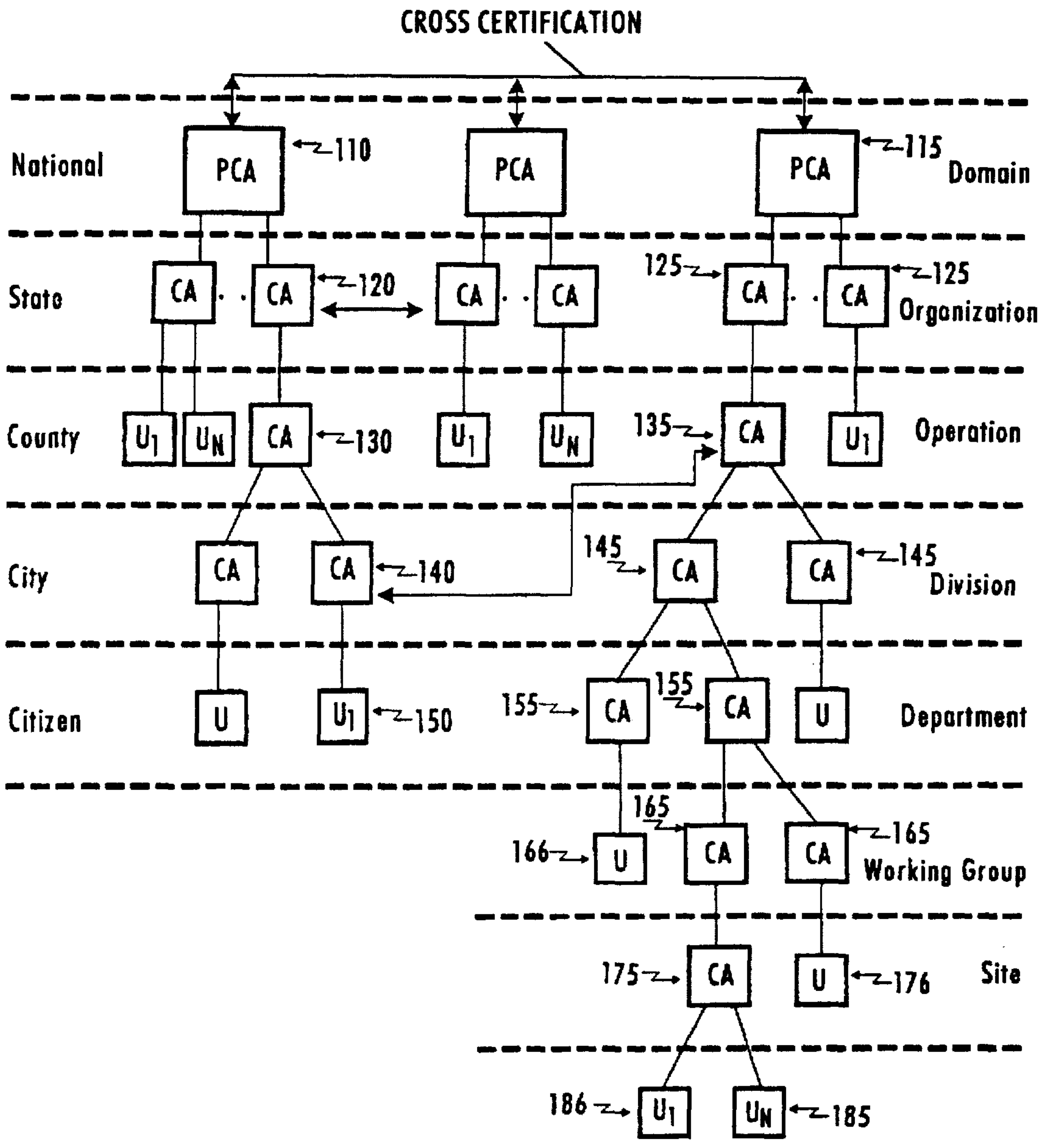
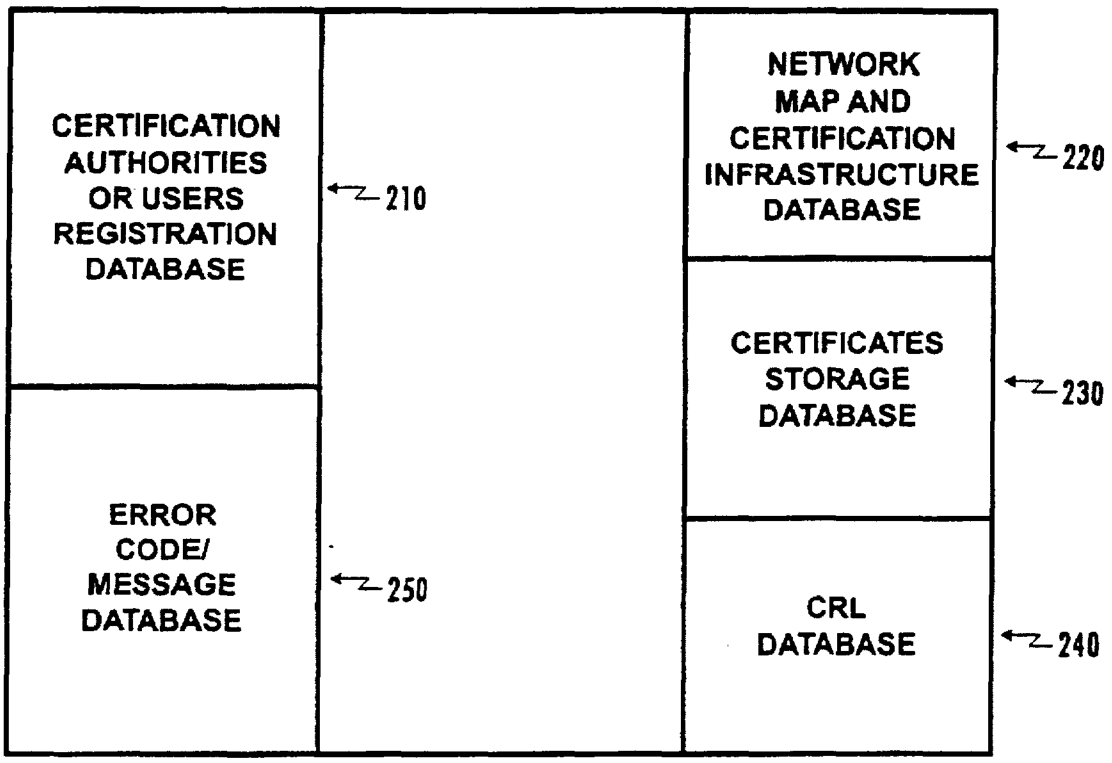

CROSS CERTIFICATION
National
PCA
110
PCA
PCA
115
Domain
State
CA
CA
120
CA
CA
125
CA
CA
125
Organization
County
U1
UN
CA
130
U1
UN
135
CA
U1
Operation
City
CA
CA
140
145
CA
CA
145
Division
Citizen
U
U1
150
155
CA
155
CA
U
Department
166
U
165
CA
CA
165
Working Group
175
CA
U
176
Site
186
U1
UN
185

Figure 1B


CERTIFICATION AUTHORITIES OR USERS REGISTRATION DATABASE
210
NETWORK MAP AND CERTIFICATION INFRASTRUCTURE DATABASE
220
CERTIFICATES STORAGE DATABASE
230
ERROR CODE/ MESSAGE DATABASE
250
CRL DATABASE
240

MEMORY/STORAGE

Figure 2

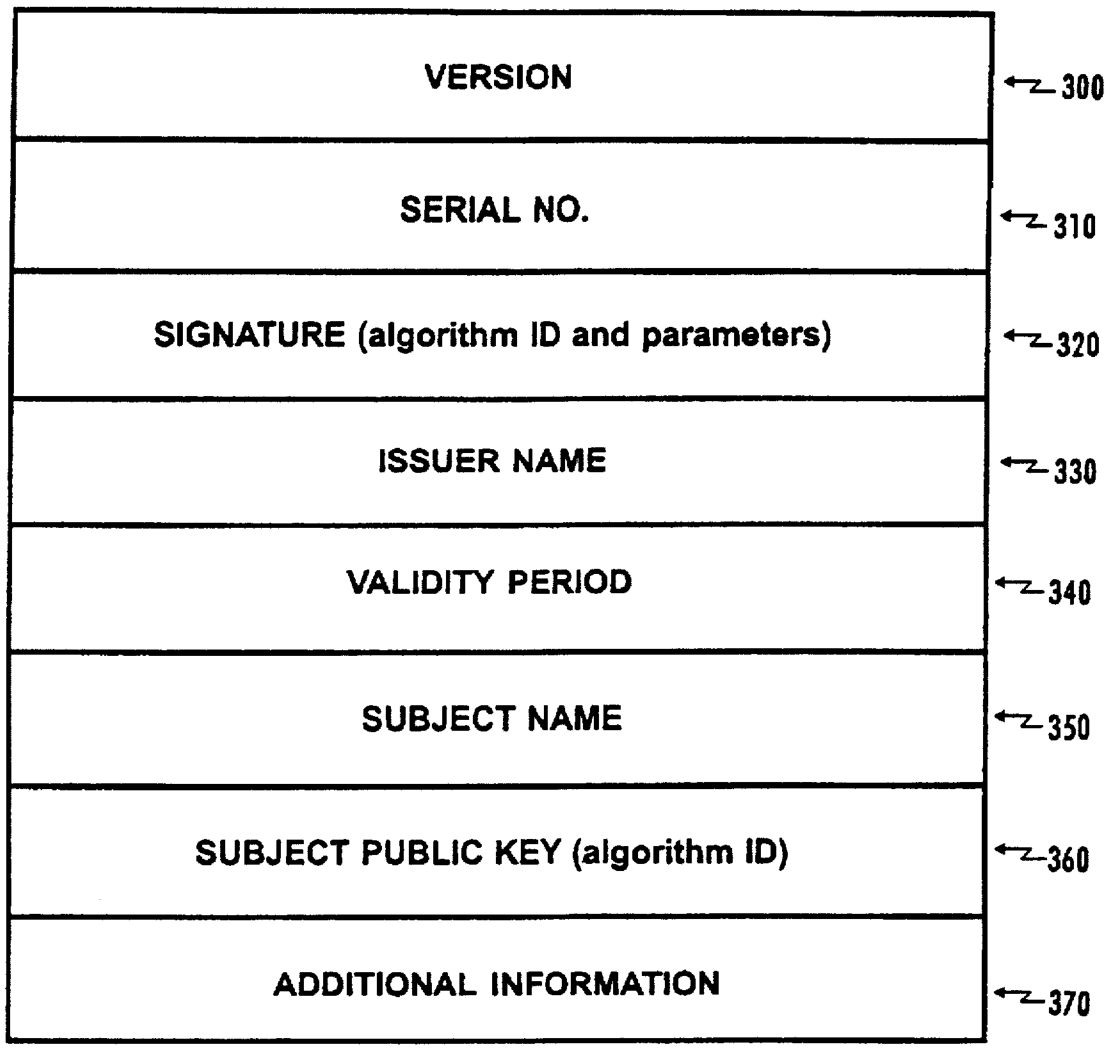

VERSION
300
SERIAL NO.
310
SIGNATURE (algorithm ID and parameters)
320
ISSUER NAME
330
VALIDITY PERIOD
340
SUBJECT NAME
350
SUBJECT PUBLIC KEY (algorithm ID)
360
ADDITIONAL INFORMATION
370

Figure 3

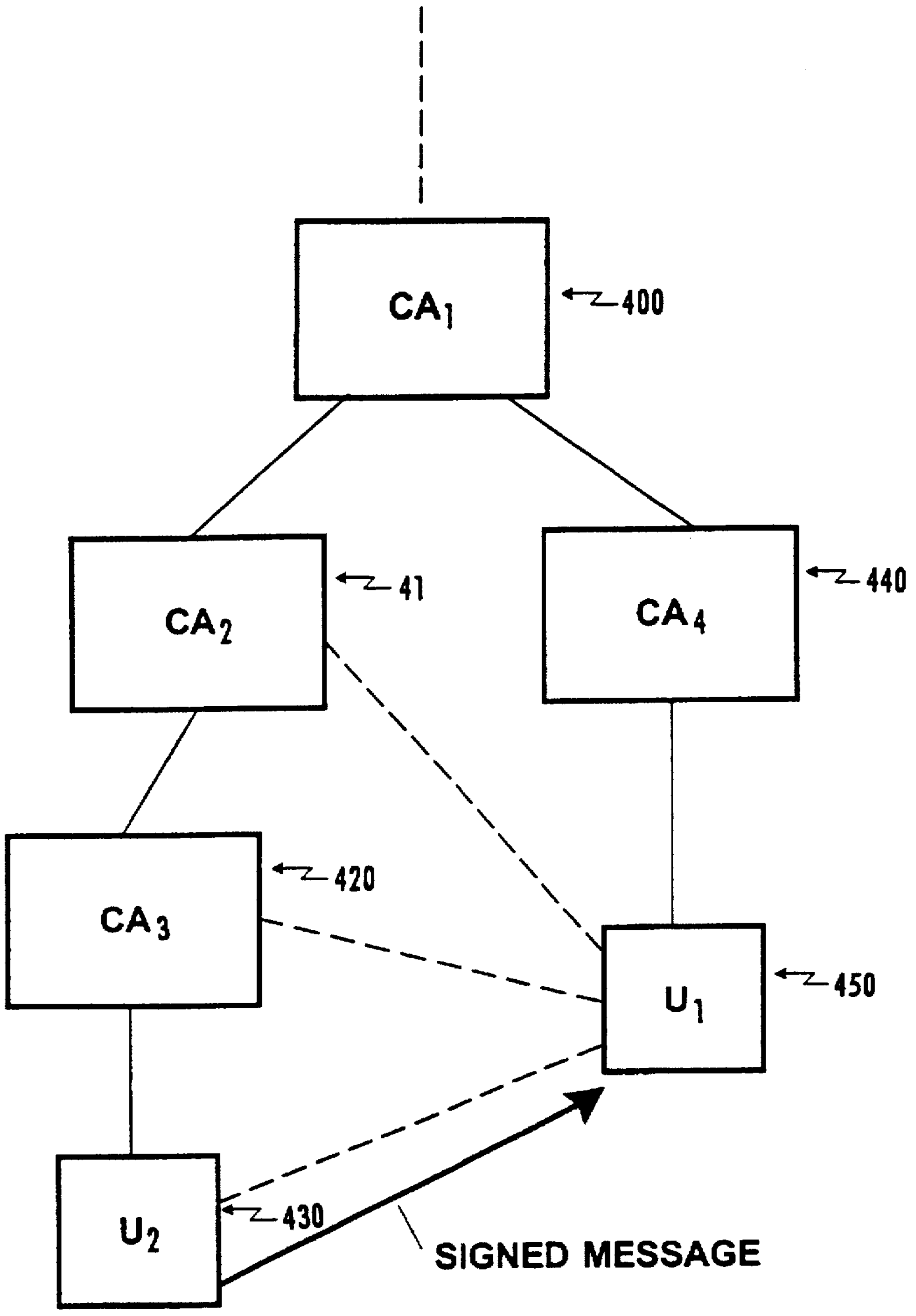


Figure 4

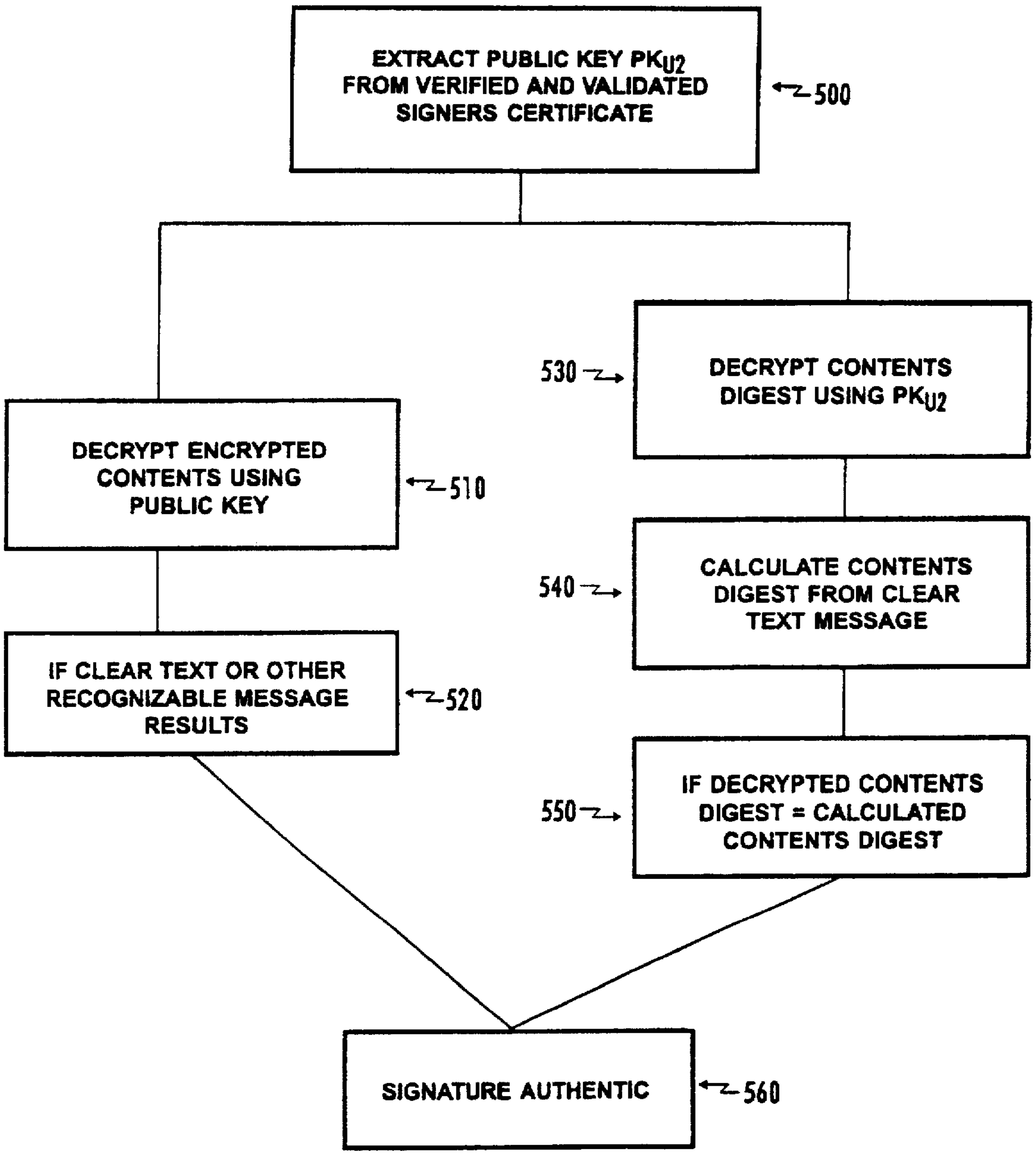

EXTRACT PUBLIC KEY PK_{U2} FROM VERIFIED AND VALIDATED SIGNERS CERTIFICATE
500
DECRYPT ENCRYPTED CONTENTS USING PUBLIC KEY
510
IF CLEAR TEXT OR OTHER RECOGNIZABLE MESSAGE RESULTS
520
530
DECRYPT CONTENTS DIGEST USING PK_{U2}
540
CALCULATE CONTENTS DIGEST FROM CLEAR TEXT MESSAGE
550
IF DECRYPTED CONTENTS DIGEST = CALCULATED CONTENTS DIGEST
SIGNATURE AUTHENTIC
560

Figure 5

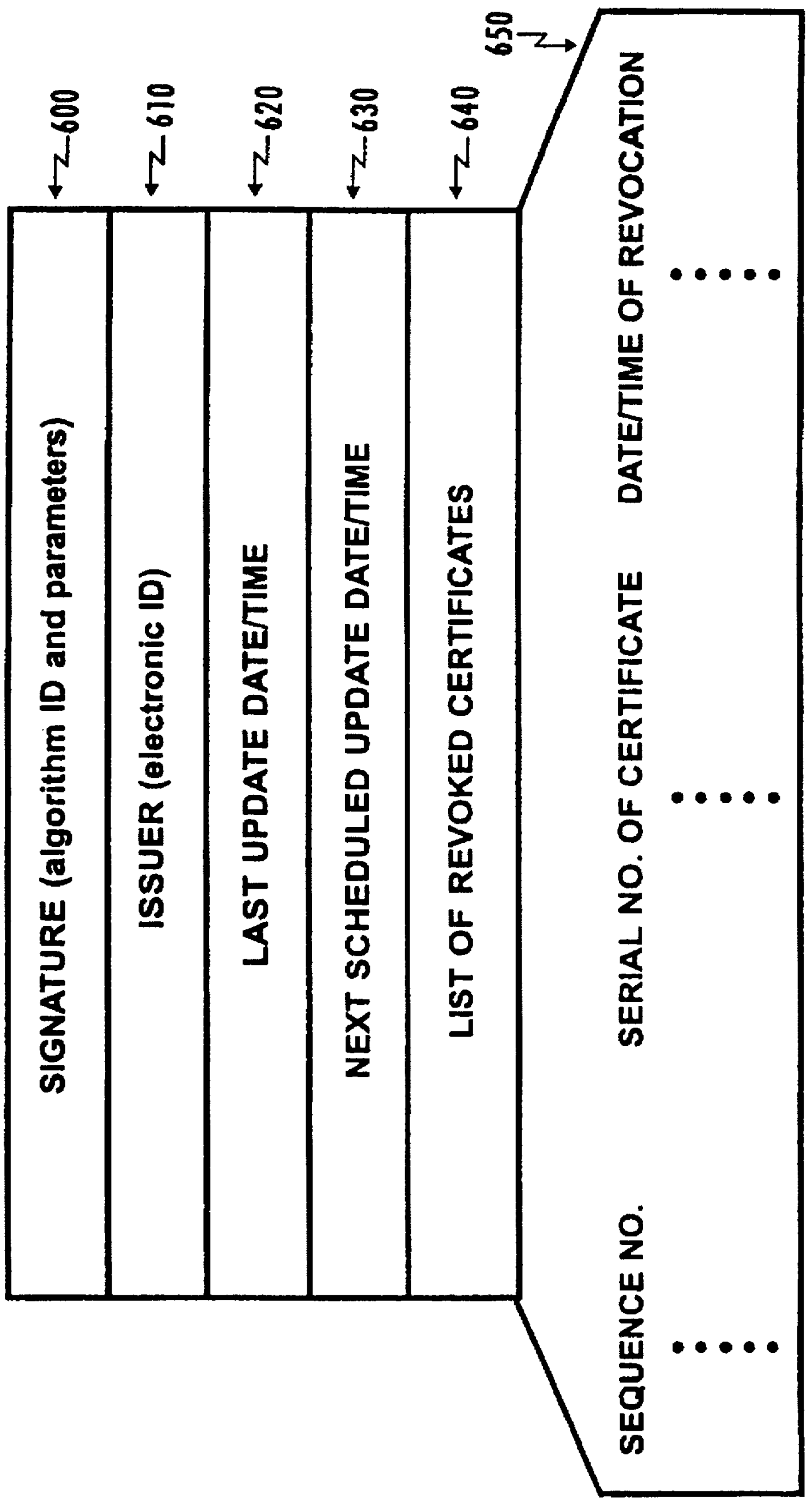

SIGNATURE (algorithm ID and parameters)
600
ISSUER (electronic ID)
610
LAST UPDATE DATE/TIME
620
NEXT SCHEDULED UPDATE DATE/TIME
630
LIST OF REVOKED CERTIFICATES
640
650
SEQUENCE NO.
SERIAL NO. OF CERTIFICATE
DATE/TIME OF REVOCATION
•••••
•••••
•••••

Figure 6

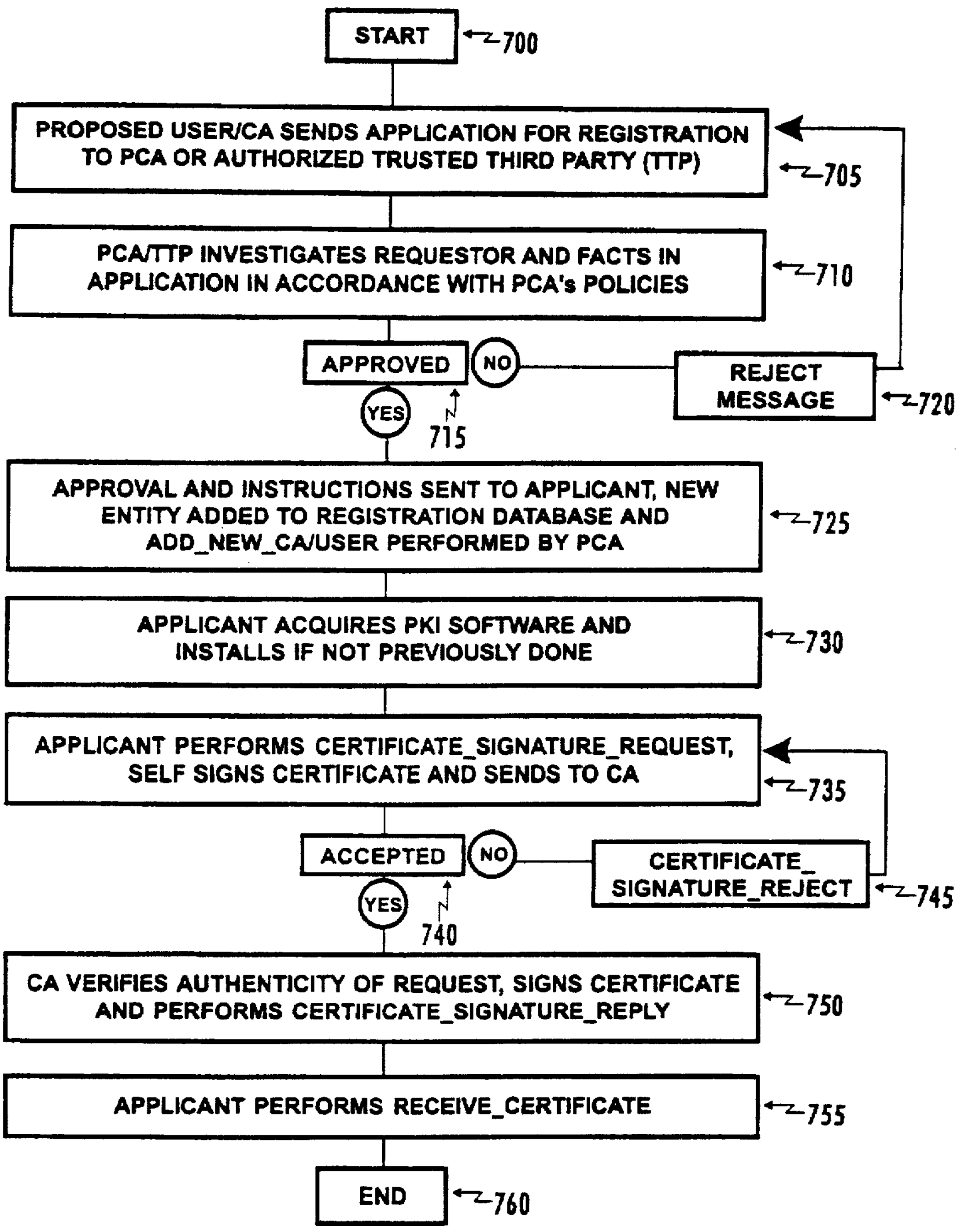

START
700
PROPOSED USER/CA SENDS APPLICATION FOR REGISTRATION TO PCA OR AUTHORIZED TRUSTED THIRD PARTY (TTP)
705
PCA/TTP INVESTIGATES REQUESTOR AND FACTS IN APPLICATION IN ACCORDANCE WITH PCA's POLICIES
710
APPROVED
NO
YES
715
REJECT MESSAGE
720
APPROVAL AND INSTRUCTIONS SENT TO APPLICANT, NEW ENTITY ADDED TO REGISTRATION DATABASE AND ADD_NEW_CA/USER PERFORMED BY PCA
725
APPLICANT ACQUIRES PKI SOFTWARE AND INSTALLS IF NOT PREVIOUSLY DONE
730
APPLICANT PERFORMS CERTIFICATE_SIGNATURE_REQUEST, SELF SIGNS CERTIFICATE AND SENDS TO CA
735
ACCEPTED
NO
YES
740
CERTIFICATE_SIGNATURE_REJECT
745
CA VERIFIES AUTHENTICITY OF REQUEST, SIGNS CERTIFICATE AND PERFORMS CERTIFICATE_SIGNATURE_REPLY
750
APPLICANT PERFORMS RECEIVE_CERTIFICATE
755
END
760

Figure 7

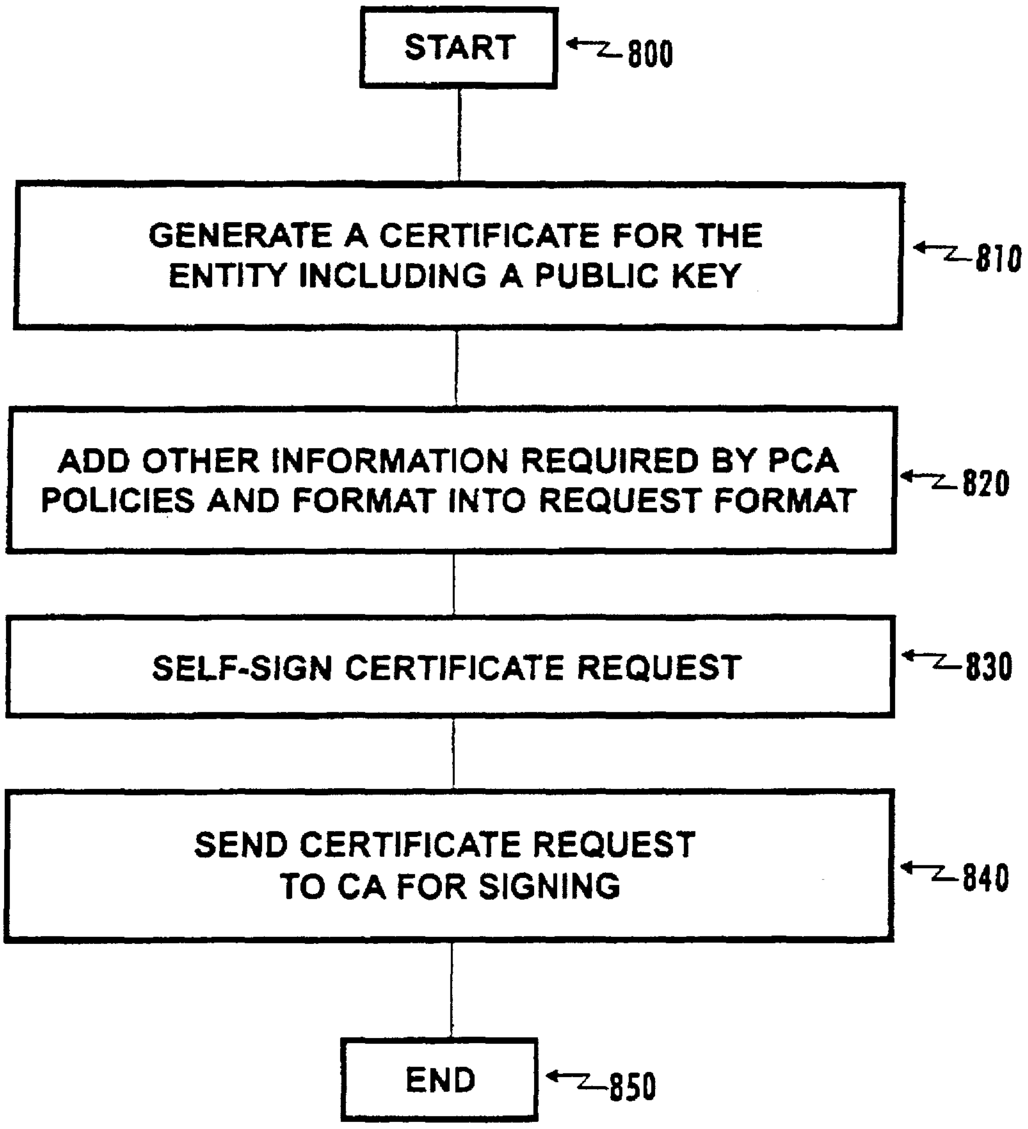

START
800
GENERATE A CERTIFICATE FOR THE ENTITY INCLUDING A PUBLIC KEY
810
ADD OTHER INFORMATION REQUIRED BY PCA POLICIES AND FORMAT INTO REQUEST FORMAT
820
SELF-SIGN CERTIFICATE REQUEST
830
SEND CERTIFICATE REQUEST TO CA FOR SIGNING
840
END
850

Figure 8

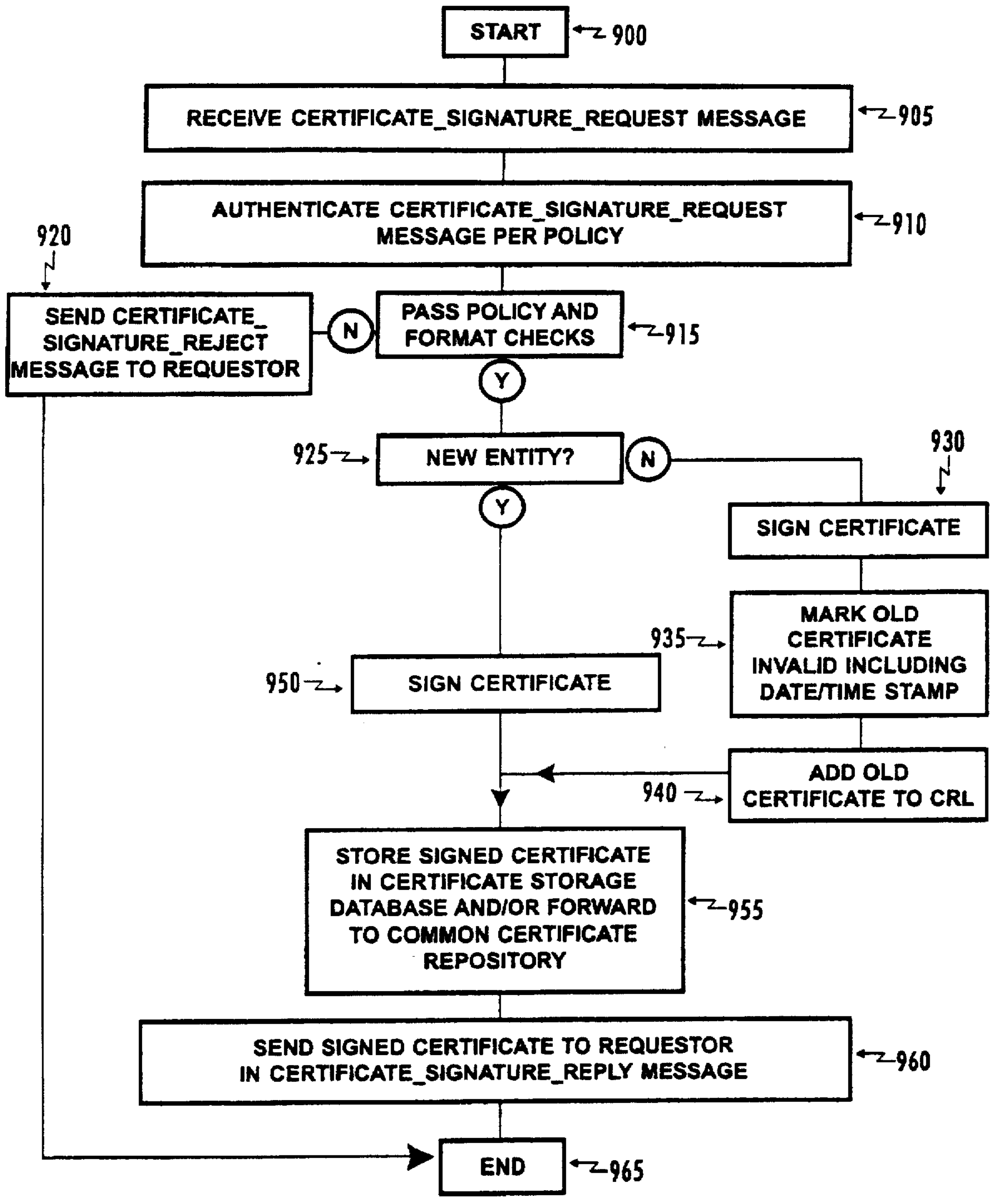

START
900
RECEIVE CERTIFICATE_SIGNATURE_REQUEST MESSAGE
905
AUTHENTICATE CERTIFICATE_SIGNATURE_REQUEST MESSAGE PER POLICY
910
920
SEND CERTIFICATE_SIGNATURE_REJECT MESSAGE TO REQUESTOR
N
PASS POLICY AND FORMAT CHECKS
915
Y
925
NEW ENTITY?
N
930
Y
SIGN CERTIFICATE
MARK OLD CERTIFICATE INVALID INCLUDING DATE/TIME STAMP
935
950
SIGN CERTIFICATE
ADD OLD CERTIFICATE TO CRL
940
STORE SIGNED CERTIFICATE IN CERTIFICATE STORAGE DATABASE AND/OR FORWARD TO COMMON CERTIFICATE REPOSITORY
955
SEND SIGNED CERTIFICATE TO REQUESTOR IN CERTIFICATE_SIGNATURE_REPLY MESSAGE
960
END
965

Figure 9

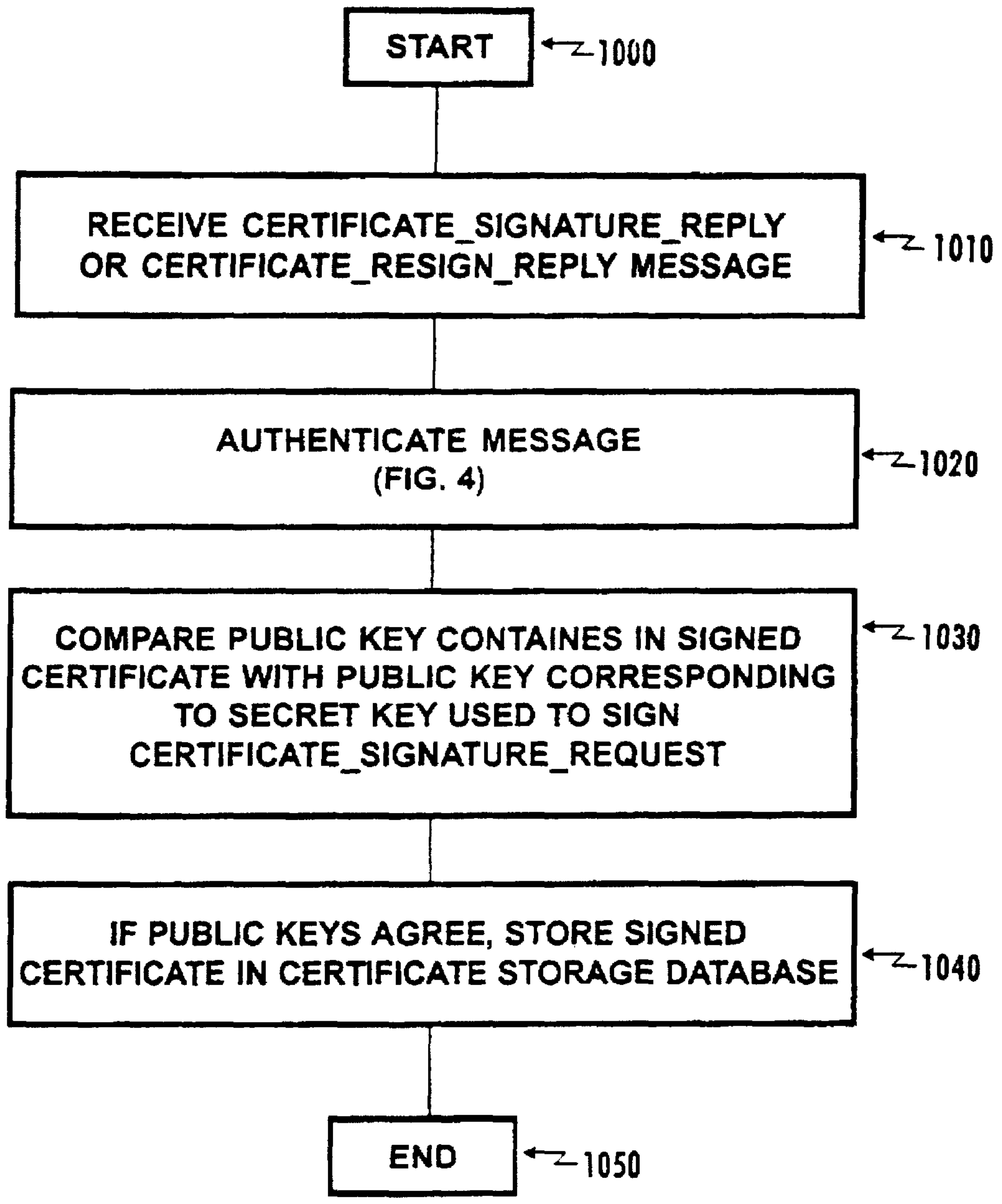

START
1000
RECEIVE CERTIFICATE_SIGNATURE_REPLY OR CERTIFICATE_RESIGN_REPLY MESSAGE
1010
AUTHENTICATE MESSAGE (FIG. 4)
1020
COMPARE PUBLIC KEY CONTAINES IN SIGNED CERTIFICATE WITH PUBLIC KEY CORRESPONDING TO SECRET KEY USED TO SIGN CERTIFICATE_SIGNATURE_REQUEST
1030
IF PUBLIC KEYS AGREE, STORE SIGNED CERTIFICATE IN CERTIFICATE STORAGE DATABASE
1040
END
1050

Figure 10

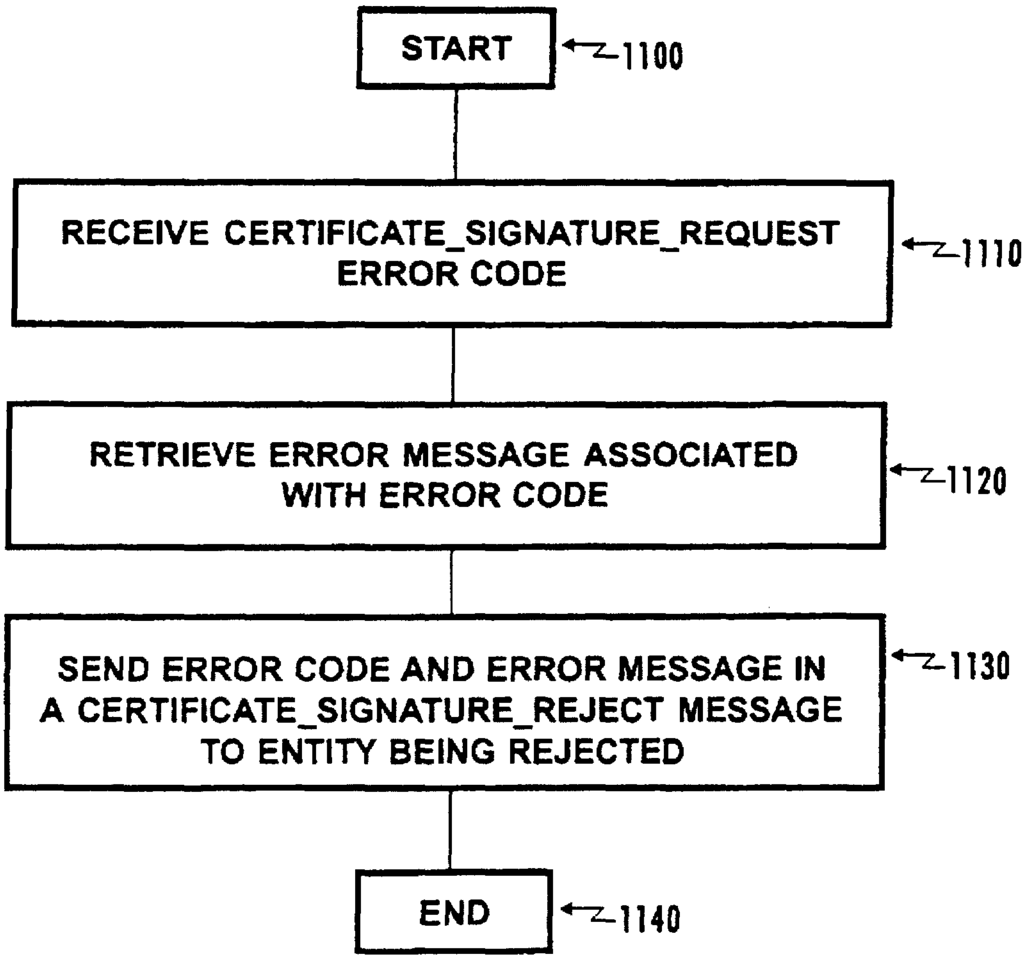

START
1100
RECEIVE CERTIFICATE_SIGNATURE_REQUEST ERROR CODE
1110
RETRIEVE ERROR MESSAGE ASSOCIATED WITH ERROR CODE
1120
SEND ERROR CODE AND ERROR MESSAGE IN A CERTIFICATE_SIGNATURE_REJECT MESSAGE TO ENTITY BEING REJECTED
1130
END
1140

Figure 11

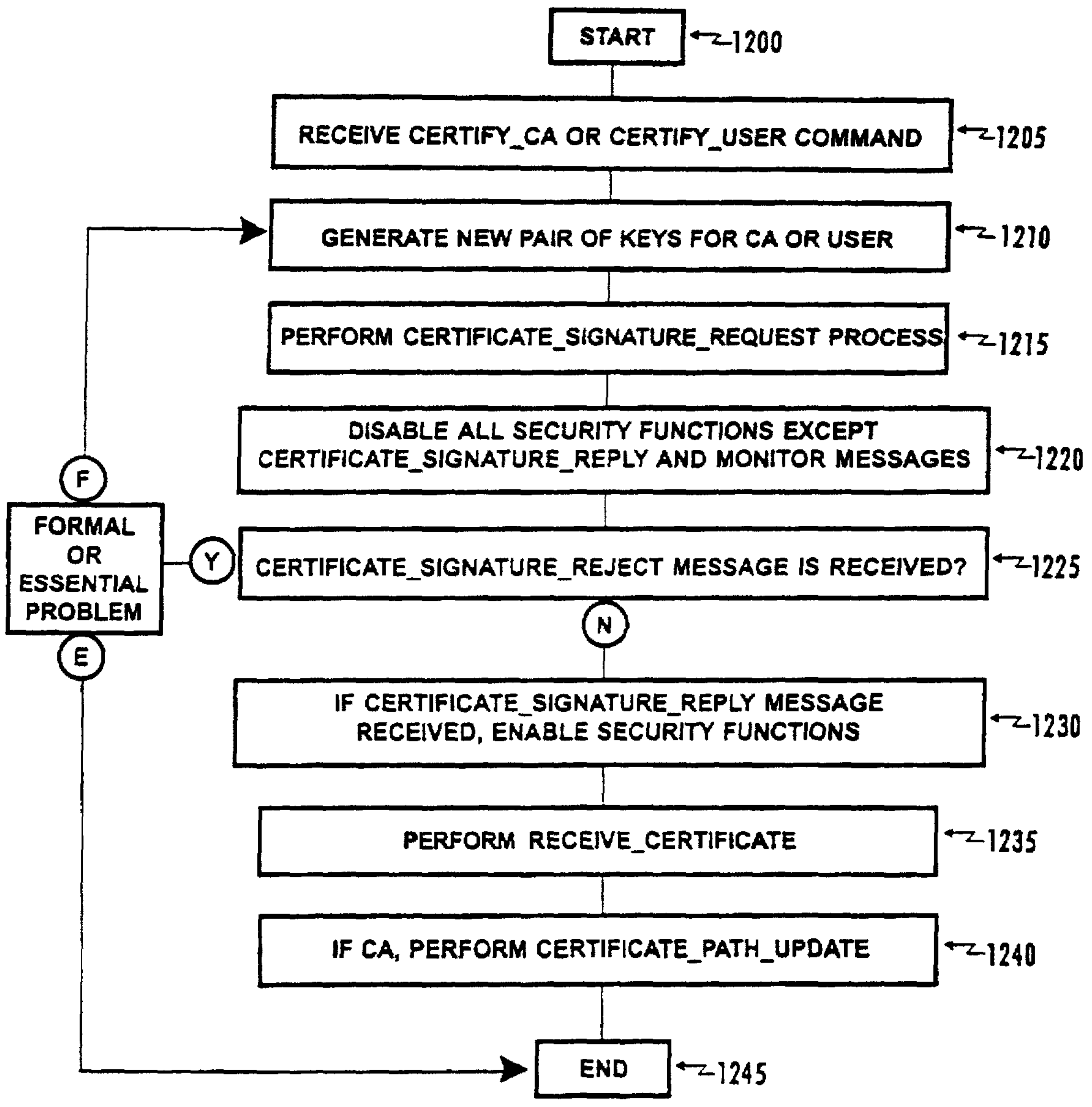

START
1200
RECEIVE CERTIFY_CA OR CERTIFY_USER COMMAND
1205
GENERATE NEW PAIR OF KEYS FOR CA OR USER
1210
PERFORM CERTIFICATE_SIGNATURE_REQUEST PROCESS
1215
DISABLE ALL SECURITY FUNCTIONS EXCEPT CERTIFICATE_SIGNATURE_REPLY AND MONITOR MESSAGES
1220
CERTIFICATE_SIGNATURE_REJECT MESSAGE IS RECEIVED?
1225
Y
N
FORMAL OR ESSENTIAL PROBLEM
F
E
IF CERTIFICATE_SIGNATURE_REPLY MESSAGE RECEIVED, ENABLE SECURITY FUNCTIONS
1230
PERFORM RECEIVE_CERTIFICATE
1235
IF CA, PERFORM CERTIFICATE_PATH_UPDATE
1240
END
1245

Figure 12

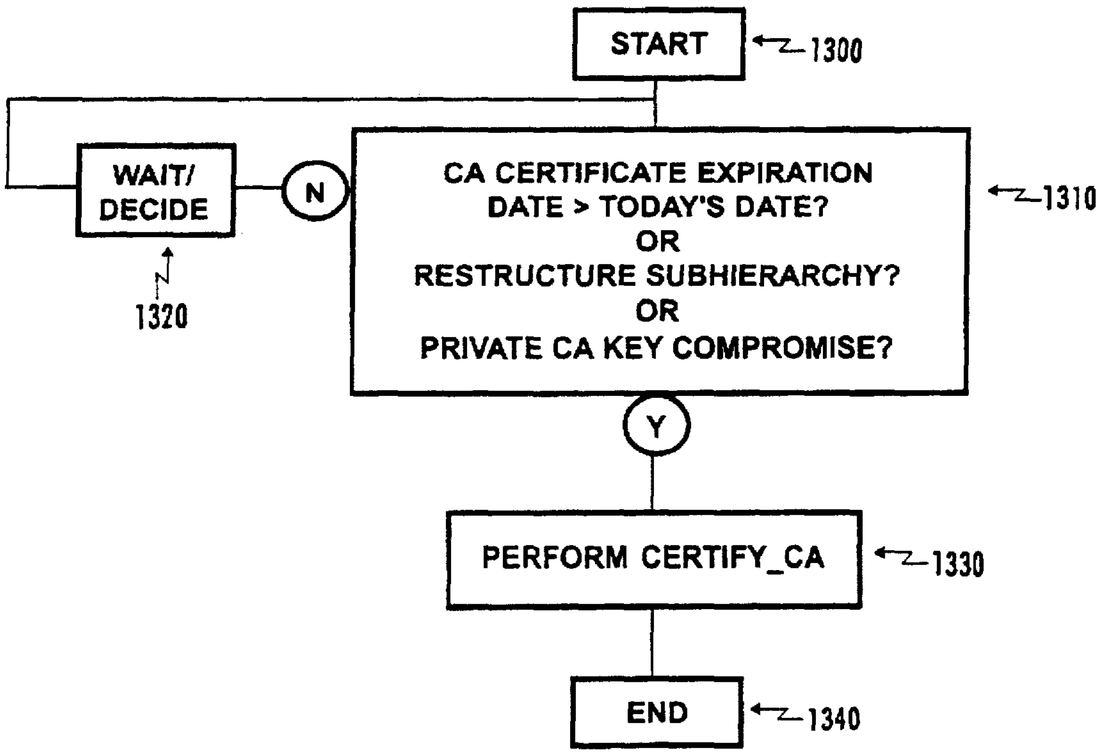

START
1300
WAIT/ DECIDE
1320
N
CA CERTIFICATE EXPIRATION DATE > TODAY'S DATE? OR RESTRUCTURE SUBHIERARCHY? OR PRIVATE CA KEY COMPROMISE?
1310
Y
PERFORM CERTIFY_CA
1330
END
1340

Figure 13

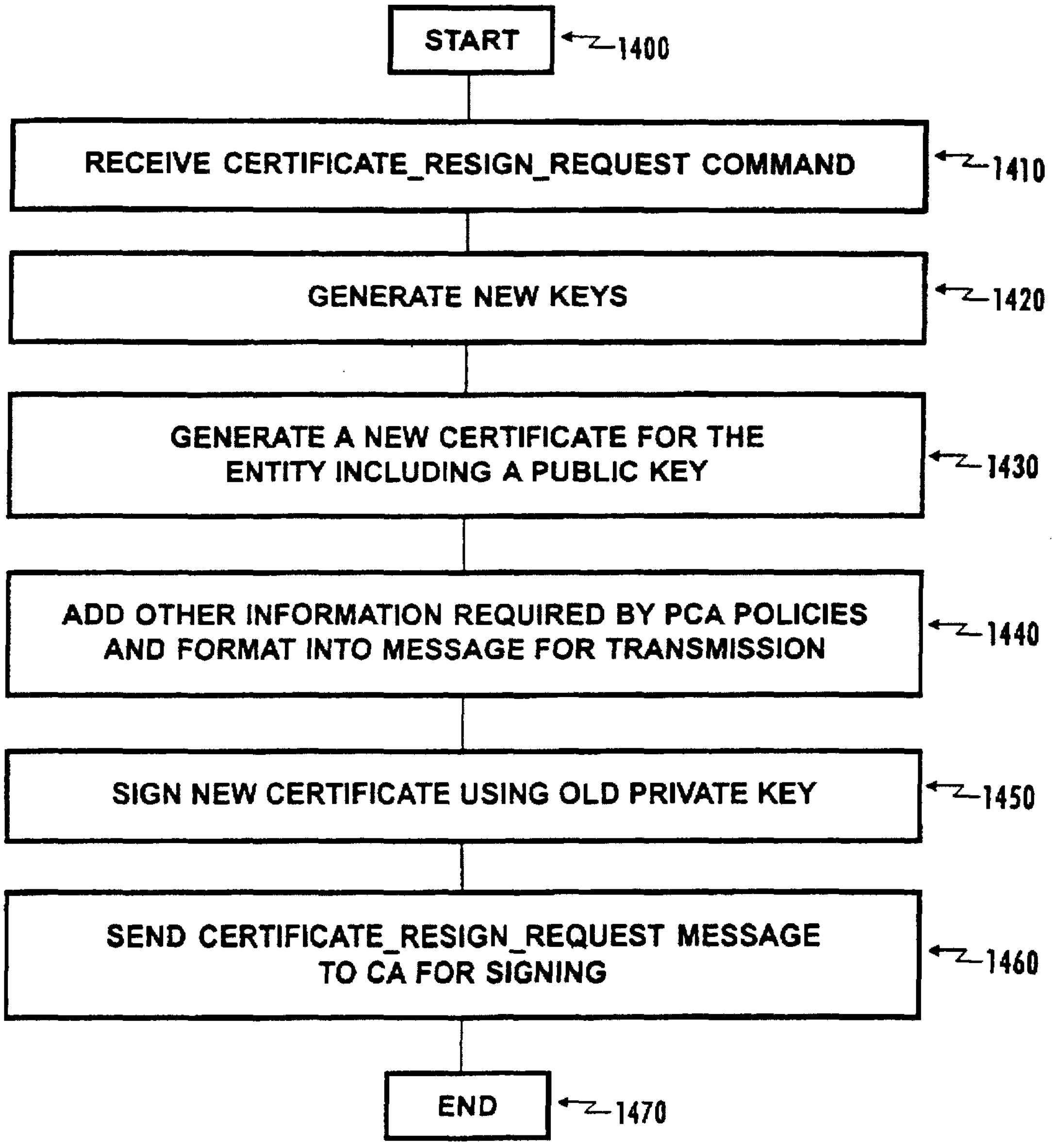

START
1400
RECEIVE CERTIFICATE_RESIGN_REQUEST COMMAND
1410
GENERATE NEW KEYS
1420
GENERATE A NEW CERTIFICATE FOR THE ENTITY INCLUDING A PUBLIC KEY
1430
ADD OTHER INFORMATION REQUIRED BY PCA POLICIES AND FORMAT INTO MESSAGE FOR TRANSMISSION
1440
SIGN NEW CERTIFICATE USING OLD PRIVATE KEY
1450
SEND CERTIFICATE_RESIGN_REQUEST MESSAGE TO CA FOR SIGNING
1460
END
1470

Figure 14

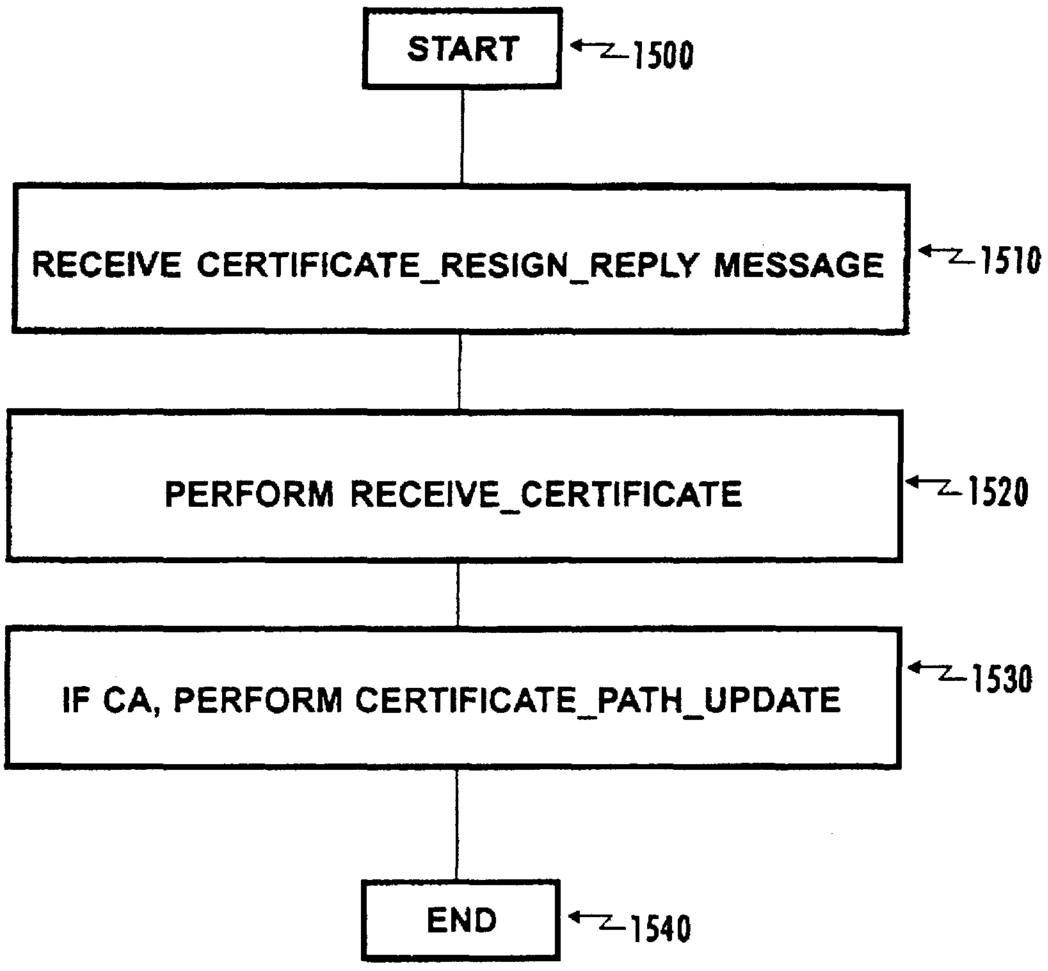

START
1500
RECEIVE CERTIFICATE_RESIGN_REPLY MESSAGE
1510
PERFORM RECEIVE_CERTIFICATE
1520
IF CA, PERFORM CERTIFICATE_PATH_UPDATE
1530
END
1540

Figure 15

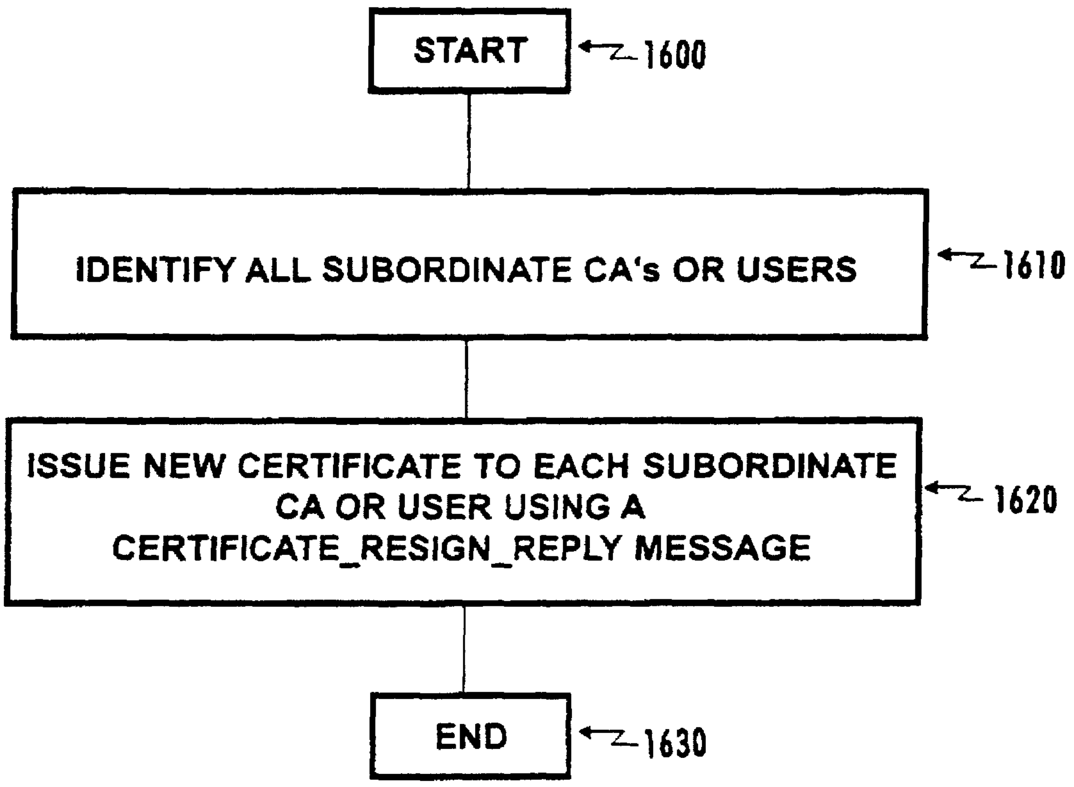

START
1600
IDENTIFY ALL SUBORDINATE CA's OR USERS
1610
ISSUE NEW CERTIFICATE TO EACH SUBORDINATE CA OR USER USING A CERTIFICATE_RESIGN_REPLY MESSAGE
1620
END
1630

Figure 16

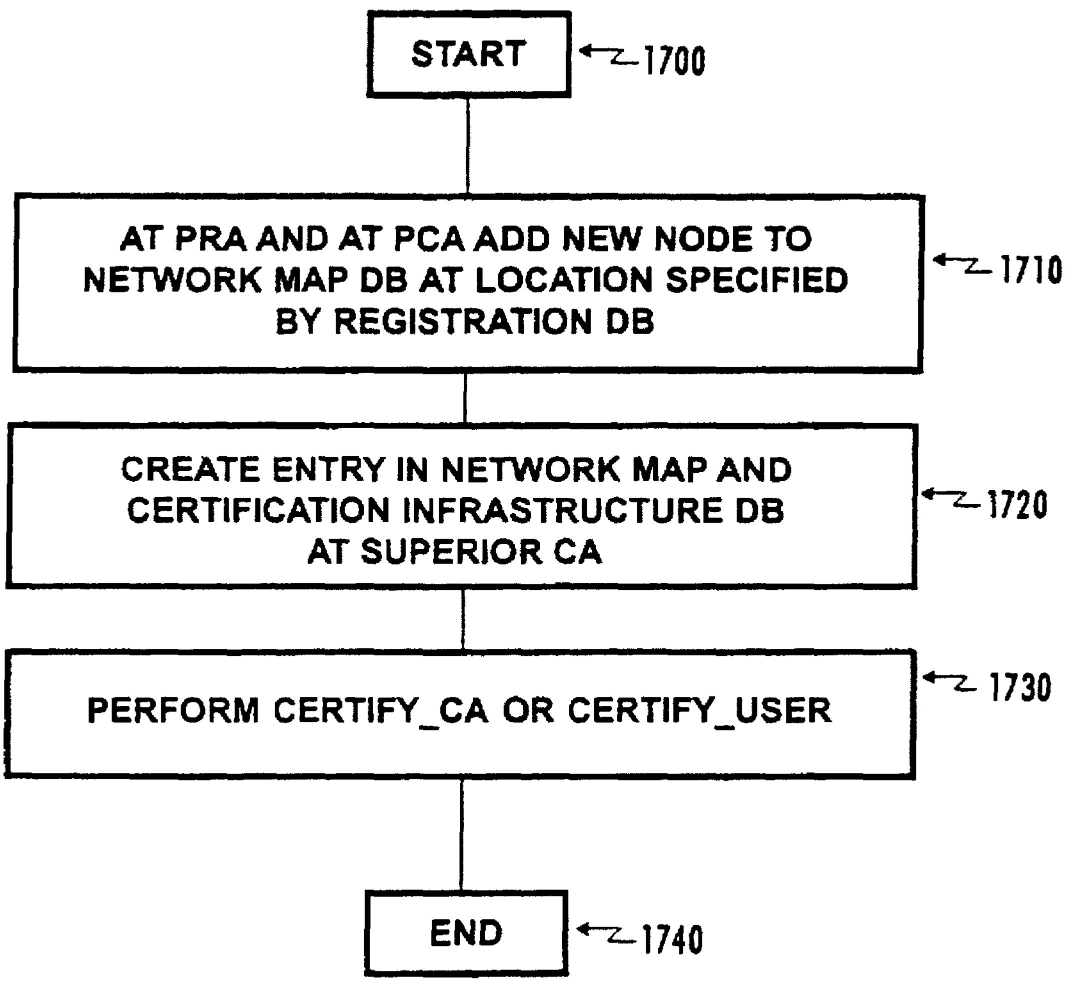

START
1700
AT PRA AND AT PCA ADD NEW NODE TO NETWORK MAP DB AT LOCATION SPECIFIED BY REGISTRATION DB
1710
CREATE ENTRY IN NETWORK MAP AND CERTIFICATION INFRASTRUCTURE DB AT SUPERIOR CA
1720
PERFORM CERTIFY_CA OR CERTIFY_USER
1730
END
1740

Figure 17

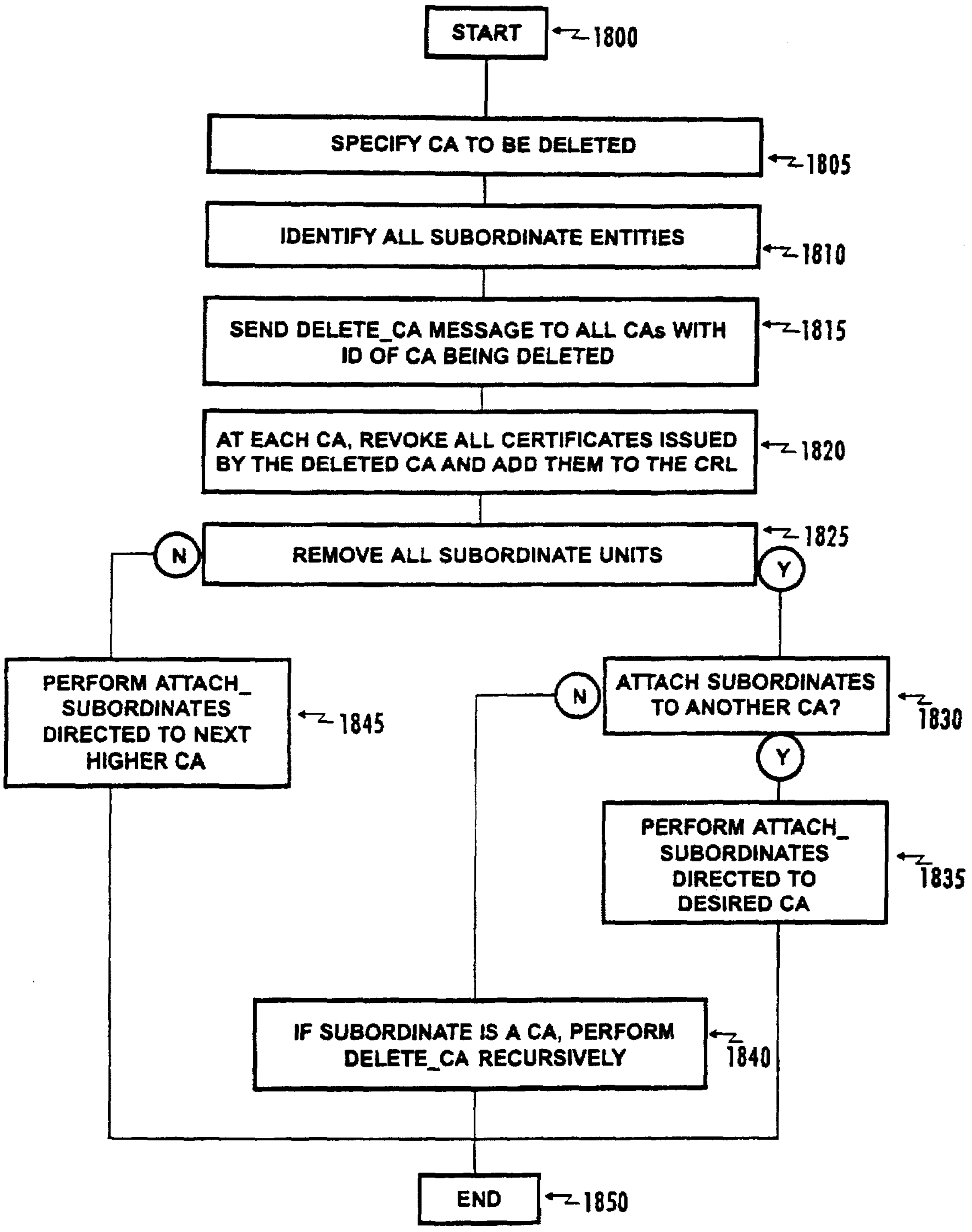

START
1800
SPECIFY CA TO BE DELETED
1805
IDENTIFY ALL SUBORDINATE ENTITIES
1810
SEND DELETE_CA MESSAGE TO ALL CAs WITH ID OF CA BEING DELETED
1815
AT EACH CA, REVOKE ALL CERTIFICATES ISSUED BY THE DELETED CA AND ADD THEM TO THE CRL
1820
REMOVE ALL SUBORDINATE UNITS
1825
N
Y
PERFORM ATTACH_ SUBORDINATES DIRECTED TO NEXT HIGHER CA
1845
ATTACH SUBORDINATES TO ANOTHER CA?
1830
N
Y
PERFORM ATTACH_ SUBORDINATES DIRECTED TO DESIRED CA
1835
IF SUBORDINATE IS A CA, PERFORM DELETE_CA RECURSIVELY
1840
END
1850

Figure 18

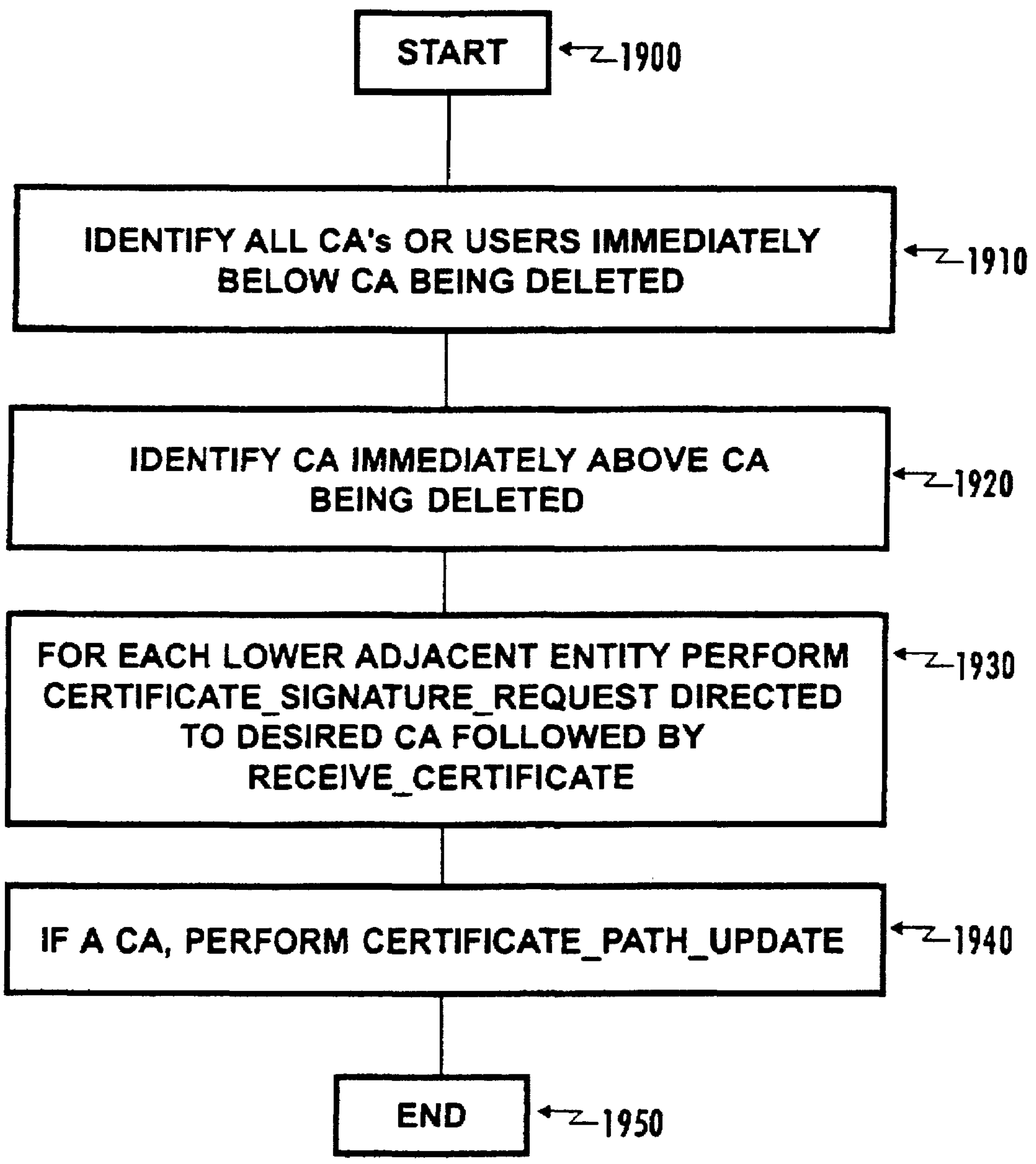

START
1900
IDENTIFY ALL CA's OR USERS IMMEDIATELY BELOW CA BEING DELETED
1910
IDENTIFY CA IMMEDIATELY ABOVE CA BEING DELETED
1920
FOR EACH LOWER ADJACENT ENTITY PERFORM CERTIFICATE_SIGNATURE_REQUEST DIRECTED TO DESIRED CA FOLLOWED BY RECEIVE_CERTIFICATE
1930
IF A CA, PERFORM CERTIFICATE_PATH_UPDATE
1940
END
1950

Figure 19

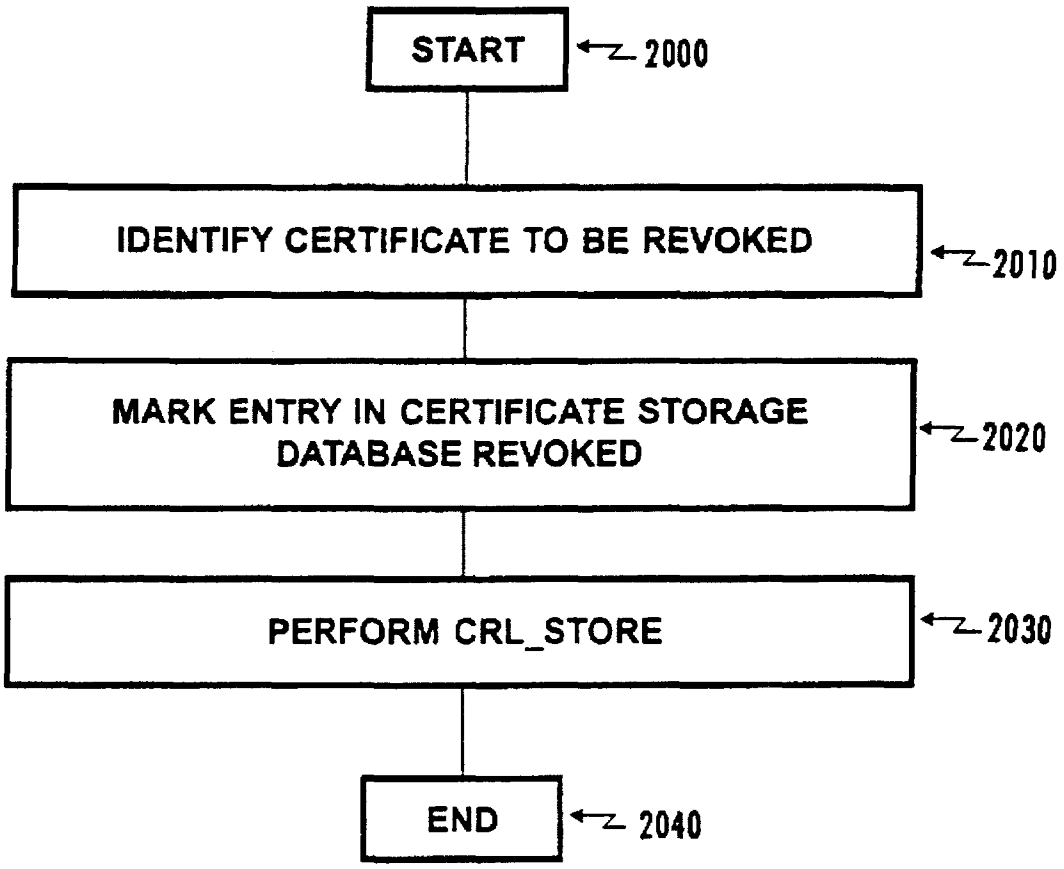

START
2000
IDENTIFY CERTIFICATE TO BE REVOKED
2010
MARK ENTRY IN CERTIFICATE STORAGE DATABASE REVOKED
2020
PERFORM CRL_STORE
2030
END
2040

Figure 20

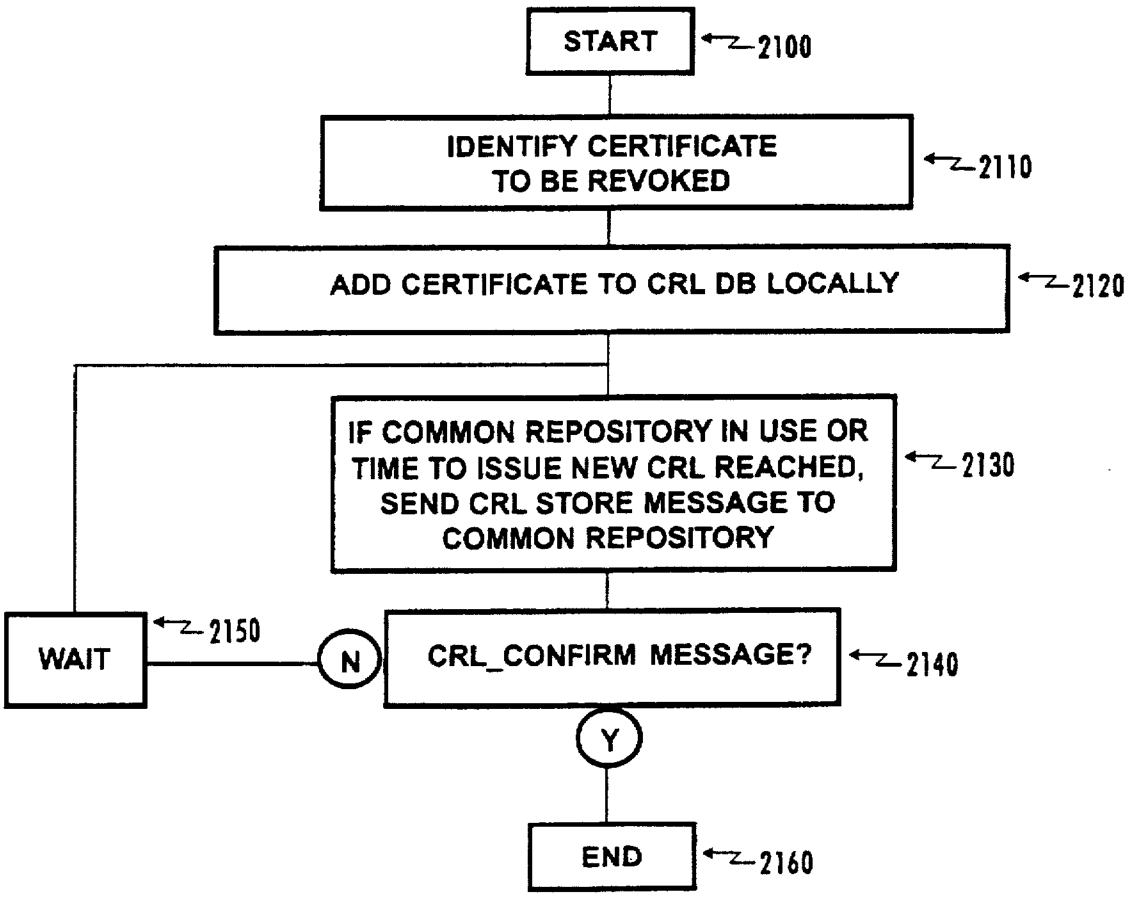

START
2100
IDENTIFY CERTIFICATE TO BE REVOKED
2110
ADD CERTIFICATE TO CRL DB LOCALLY
2120
IF COMMON REPOSITORY IN USE OR TIME TO ISSUE NEW CRL REACHED, SEND CRL STORE MESSAGE TO COMMON REPOSITORY
2130
WAIT
2150
N
CRL_CONFIRM MESSAGE?
2140
Y
END
2160

Figure 21

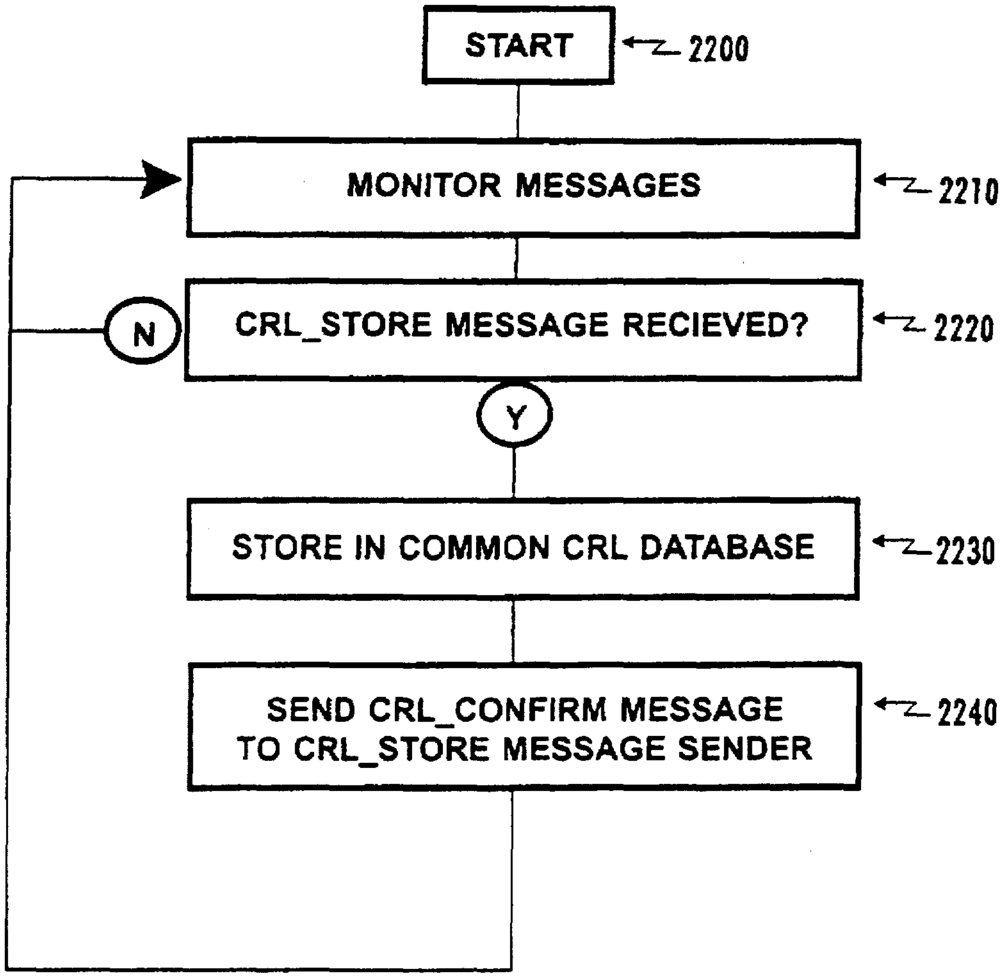

START
2200
MONITOR MESSAGES
2210
CRL_STORE MESSAGE RECIEVED?
2220
N
Y
STORE IN COMMON CRL DATABASE
2230
SEND CRL_CONFIRM MESSAGE TO CRL_STORE MESSAGE SENDER
2240

Figure 22

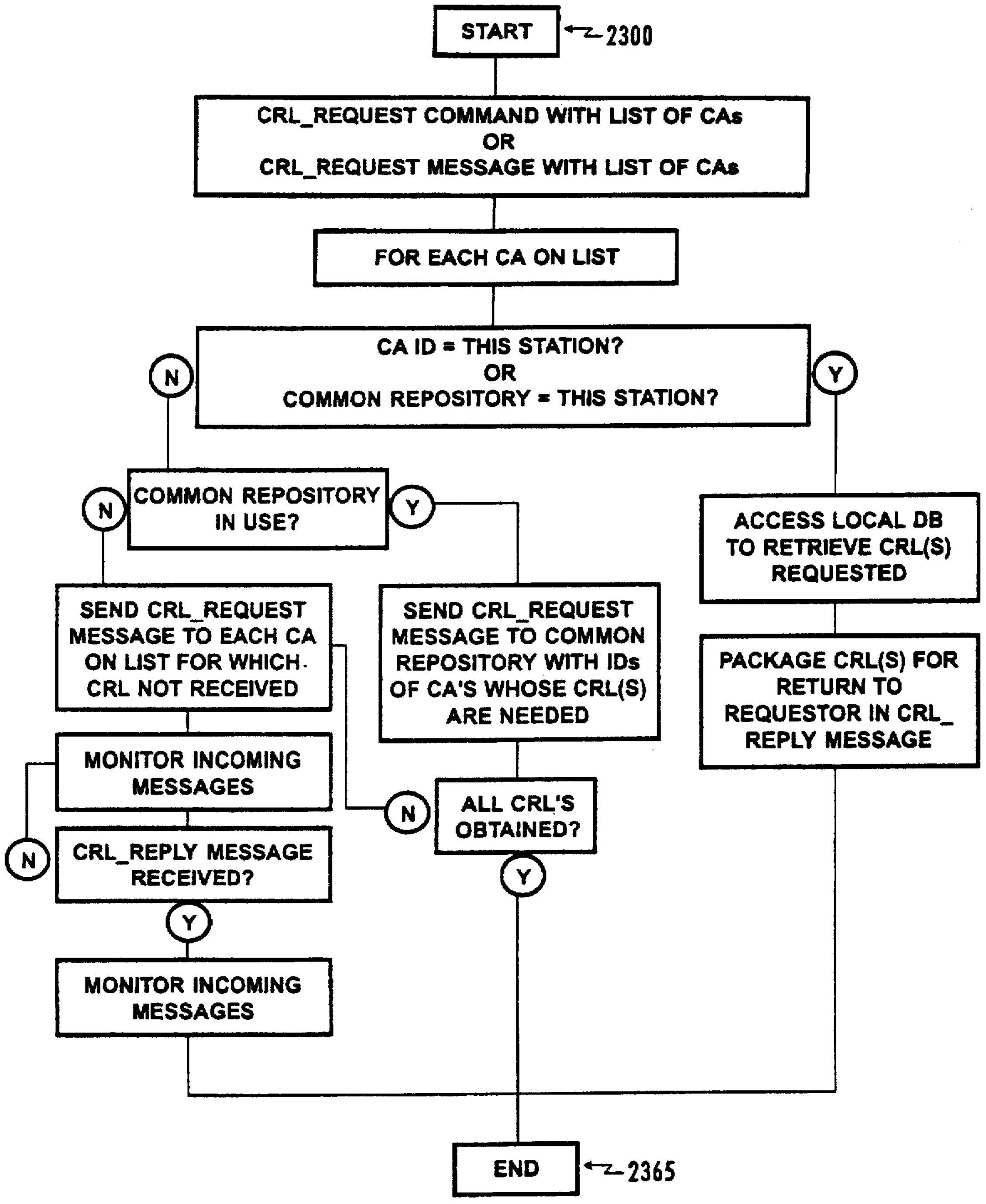

START
2300
CRL_REQUEST COMMAND WITH LIST OF CAs OR CRL_REQUEST MESSAGE WITH LIST OF CAs
FOR EACH CA ON LIST
CA ID = THIS STATION? OR COMMON REPOSITORY = THIS STATION?
N
Y
COMMON REPOSITORY IN USE?
N
Y
ACCESS LOCAL DB TO RETRIEVE CRL(S) REQUESTED
SEND CRL_REQUEST MESSAGE TO EACH CA ON LIST FOR WHICH CRL NOT RECEIVED
SEND CRL_REQUEST MESSAGE TO COMMON REPOSITORY WITH IDs OF CA'S WHOSE CRL(S) ARE NEEDED
PACKAGE CRL(S) FOR RETURN TO REQUESTOR IN CRL_ REPLY MESSAGE
MONITOR INCOMING MESSAGES
N
ALL CRL'S OBTAINED?
N
CRL_REPLY MESSAGE RECEIVED?
Y
Y
MONITOR INCOMING MESSAGES
END
2365

Figure 23

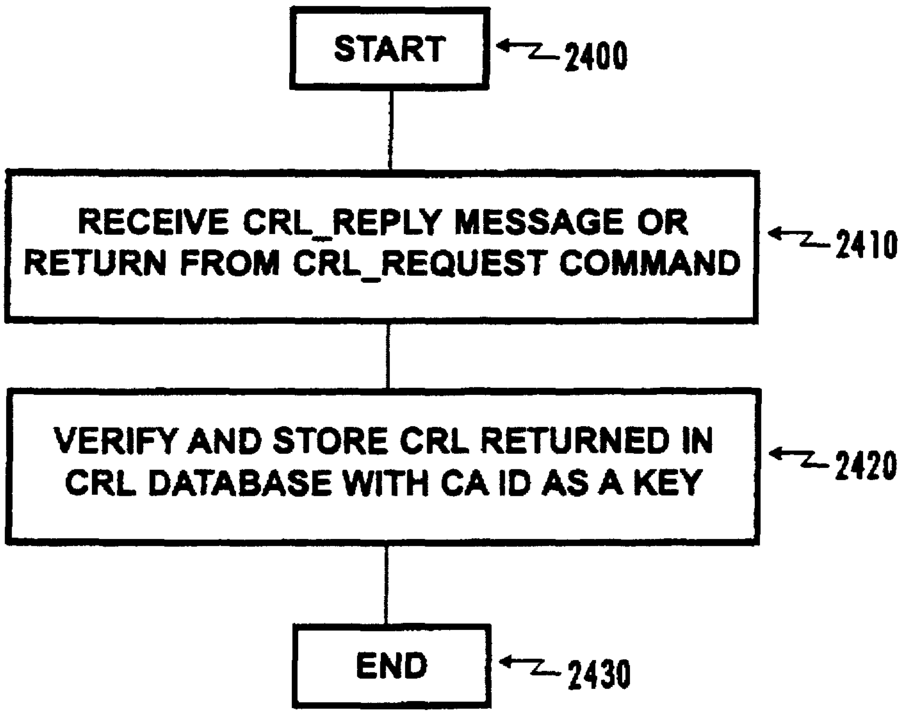

START
2400
RECEIVE CRL_REPLY MESSAGE OR RETURN FROM CRL_REQUEST COMMAND
2410
VERIFY AND STORE CRL RETURNED IN CRL DATABASE WITH CA ID AS A KEY
2420
END
2430

Figure 24

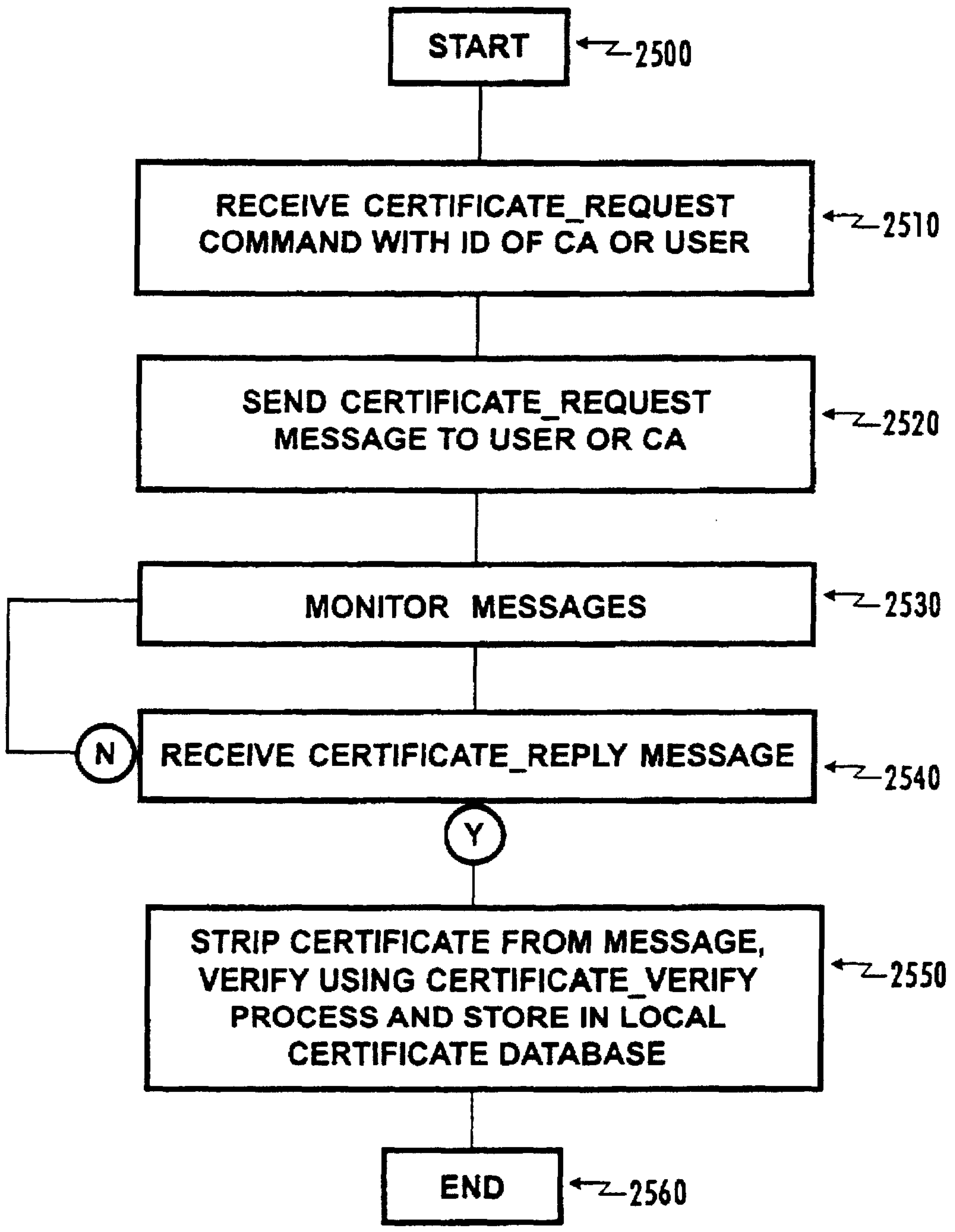

START
2500
RECEIVE CERTIFICATE_REQUEST COMMAND WITH ID OF CA OR USER
2510
SEND CERTIFICATE_REQUEST MESSAGE TO USER OR CA
2520
MONITOR MESSAGES
2530
RECEIVE CERTIFICATE_REPLY MESSAGE
N
Y
2540
STRIP CERTIFICATE FROM MESSAGE, VERIFY USING CERTIFICATE_VERIFY PROCESS AND STORE IN LOCAL CERTIFICATE DATABASE
2550
END
2560

Figure 25

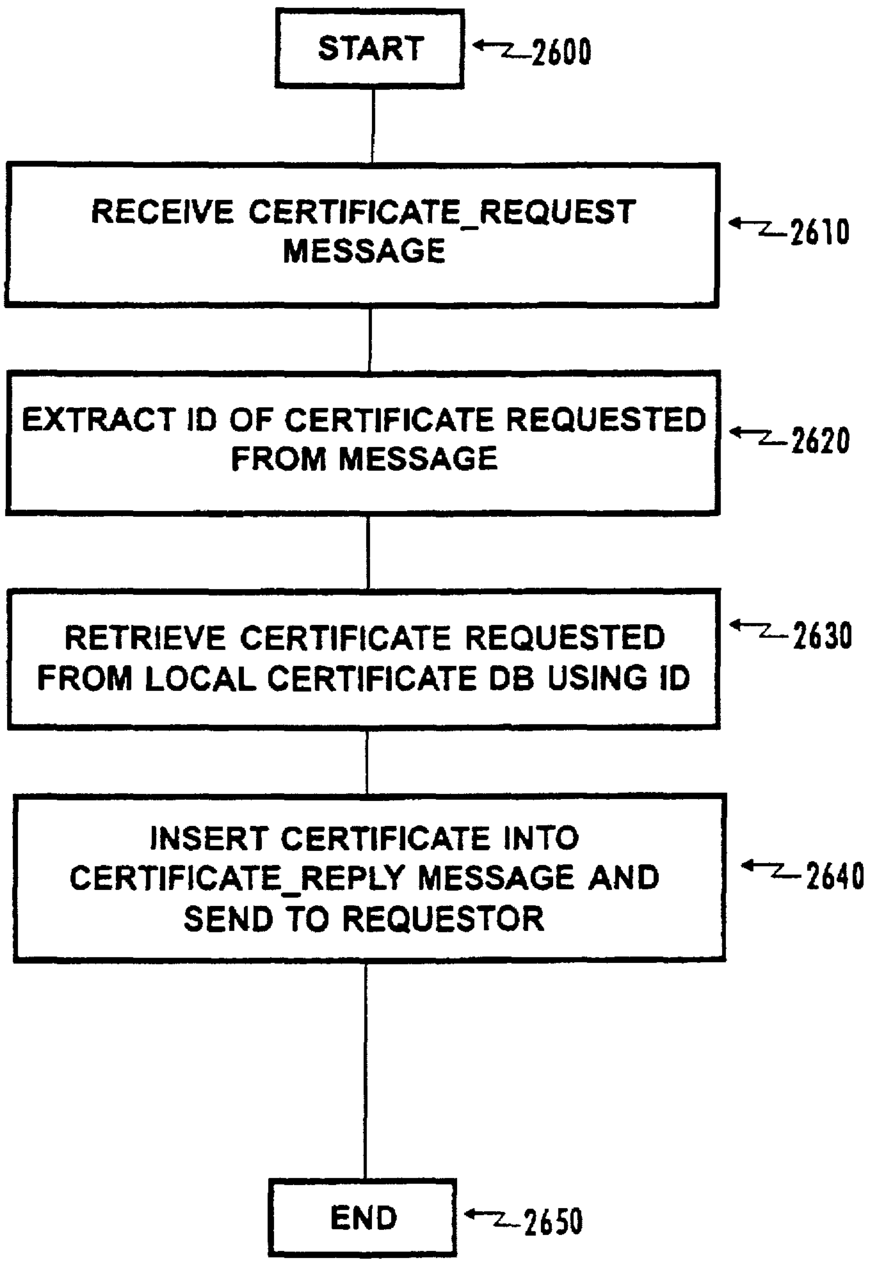

START
2600
RECEIVE CERTIFICATE_REQUEST MESSAGE
2610
EXTRACT ID OF CERTIFICATE REQUESTED FROM MESSAGE
2620
RETRIEVE CERTIFICATE REQUESTED FROM LOCAL CERTIFICATE DB USING ID
2630
INSERT CERTIFICATE INTO CERTIFICATE_REPLY MESSAGE AND SEND TO REQUESTOR
2640
END
2650

Figure 26

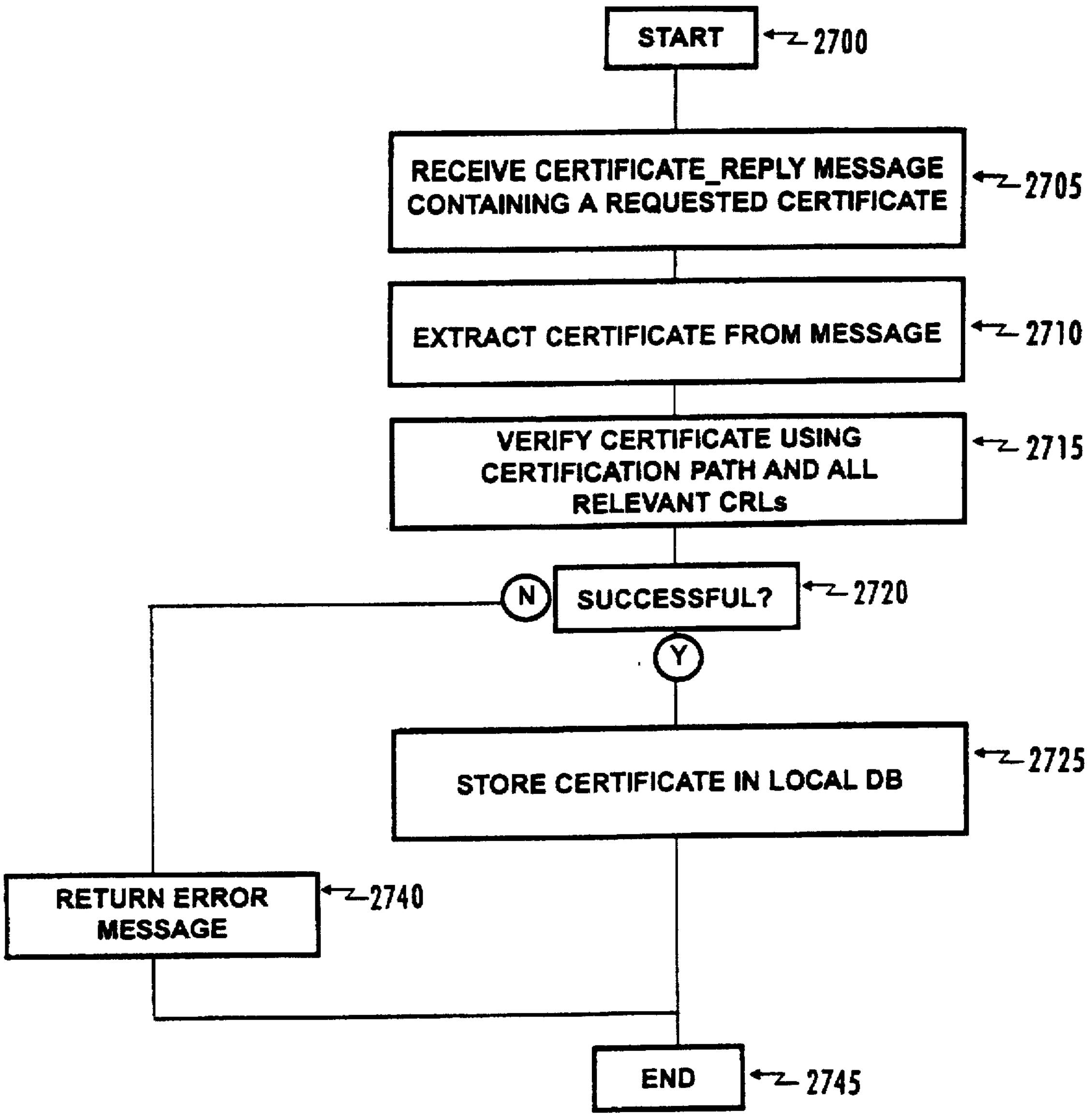

START
2700
RECEIVE CERTIFICATE_REPLY MESSAGE CONTAINING A REQUESTED CERTIFICATE
2705
EXTRACT CERTIFICATE FROM MESSAGE
2710
VERIFY CERTIFICATE USING CERTIFICATION PATH AND ALL RELEVANT CRLs
2715
SUCCESSFUL?
2720
N
Y
STORE CERTIFICATE IN LOCAL DB
2725
RETURN ERROR MESSAGE
2740
END
2745

Figure 27

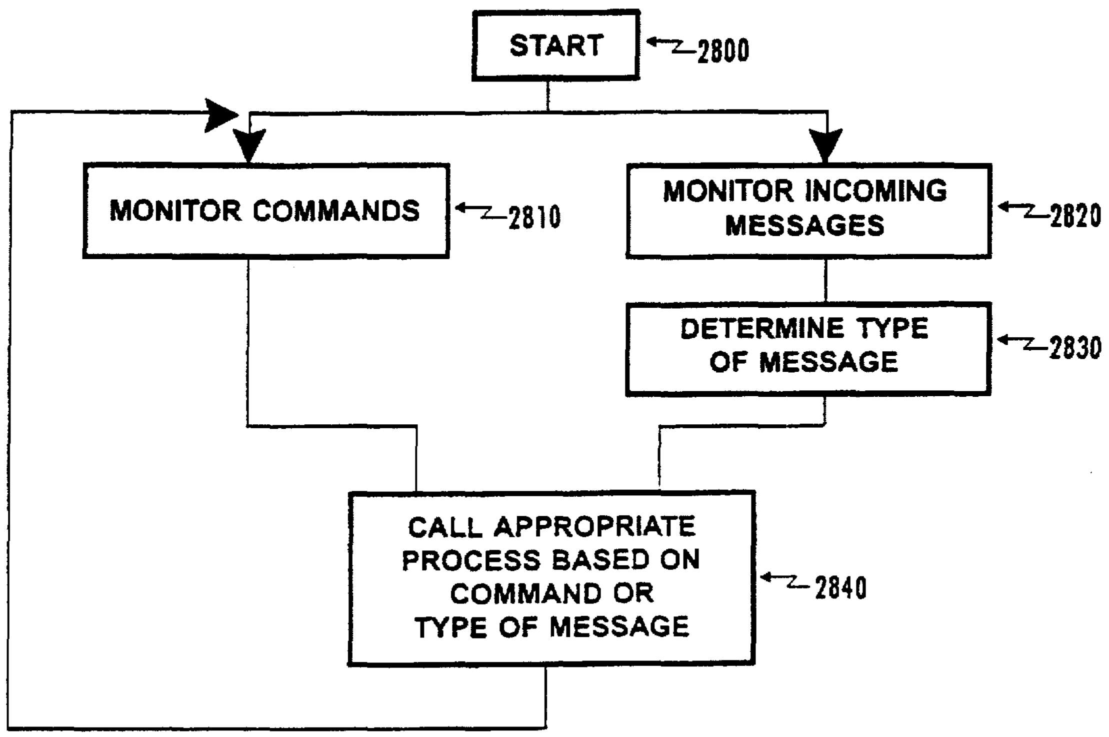

START
2800
MONITOR COMMANDS
2810
MONITOR INCOMING MESSAGES
2820
DETERMINE TYPE OF MESSAGE
2830
CALL APPROPRIATE PROCESS BASED ON COMMAND OR TYPE OF MESSAGE
2840

Figure 28

SECURITY INFRASTRUCTURE FOR ELECTRONIC TRANSACTIONS

TECHNICAL FIELD

The invention is directed to computer communication systems and more particularly to public key encryption based secure communication systems.

BACKGROUND ART

Encryption of information is normally undertaken to ensure privacy, that is, so that no one other than the intended recipient can decipher the information. Encryption is also undertaken to ensure the authenticity of the information, that is, that a message which purports to originate with a particular source actually and has not been tampered with.

"Encrypting" or "enciphering" a message means to scramble it in a way which renders it unreadable to anyone except the intended recipient(s). In one form, a cryptographic "key" is utilized to encrypt the message and the same key is required to transform it from encrypted form back to plain text by deciphering or decrypting it. An encryption system which operates in this way is known as a "single-key" encryption system. In such a system, the key must be available to both the sender and the receiver. If unauthorized persons have access to the key, then they can decrypt the encoded message and the object of privacy is defeated. The most obvious drawback of single key encryption systems is that it is not often convenient to provide the sender and the receiver with keys. They may be located far apart. A key can be transmitted across a secure channel from the sender to the receiver, but if a secure channel is available, there is no need for encryption.

In a public key encryption system each participant has two related keys. A public key which is publicly available and a related private key or secret key which is not. The public and private keys are duals of each other in the sense that material encrypted with the public key can only be decrypted using the private key. Material encrypted with the private key, on the other hand, can be decrypted only using the public key. The keys utilized in public key encryption systems are such that information about the public key does not help deduce the corresponding private key. The public key can be published and widely disseminated across a communications network or otherwise and material can be sent in privacy to a recipient by encrypting the material with the recipient's public key. Only the recipient can decrypt material encrypted with the recipient's public key. Not even the originator who does the encryption using the recipient's public key is able to decrypt that which he himself has encrypted.

Message authentication can also be achieved utilizing encryption systems. In a single key system, a sender, by encrypting a message with a key known only to authorized persons, tells the recipient that the message came from an authorized source.

In a public key encryption system, if the sender encrypts information using the sender's private key, all recipients will be able to decipher the information using the sender's public key, which is available to all. The recipients can be assured that the information originated with the sender, because the public key will only decrypt material encrypted with the sender's private key. Since presumably, only the sender has the private key, the sender cannot later disavow that he sent the information.

The use of encryption techniques provides a basis for creating electronic signatures to documents which are even

less subject to forgery than handwritten signatures. There are two ways in which encryption can be utilized to "sign" a document. The first method is by encrypting the entire document using the signer's private key. The document can be read by anyone with the signer's public key and, since the signer alone possesses his private key, the encrypted document surely originated with the signer. Encryption of large documents requires considerable computational resources and, to speed up the process, a message digest may be used.

A message digest of the document is analogous to a cyclic redundancy code (CRC) check sum attached to the end of a packet. The information in the body of the packet is processed mathematically to produce a unique check sum which is appended to the end of the packet. The integrity of the body of the packet is checked at the receiving end by recalculating the check sum based on the received text and verifying if it matches the check sum appended to the packet. If it does, one assumes that the contents of the body of packet is unchanged from that present at the sending end. The same can be done with entire documents.

In modern implementations, a message digest is created using a cryptographically strong one way hash function between the message text and the output digest and the message digest operates like a CRC check sum.

A clear text document may be signed by creating the message digest and then by encrypting the message digest using the signer's private key. Authentication that the content of the document has not been changed is achieved by computing the same one way hash function of the received text, from the text, and comparing it with the message digest decrypted using the signer's public key. If they agree, one may have a high degree of confidence that the document has been unchanged from the time it was signed, until the present and further, that that which the sender "signed" was the same document.

Public key encryption software is widely available. For example, Pretty Good™ Privacy public key encryption software is available for non-commercial use over the Internet in a form published by Phillip Zimmerman. One version, is PGP version 2.6.2 of Oct. 11, 1994. It is available from the Massachusetts Institute of Technology at net-dis.mit.adu, a controlled FTP site that has restrictions and limitations to comply with export control requirements. Software resides in the directory /pub/PGP. A fully licensed version of PGP for commercial use in the U.S.A. and Canada is available through ViaCrypt in Pheonix, Ariz.

Some public key encryption systems utilize a single key encryption of the body of the text with the key changing from session to session and with the key encrypted utilizing the recipient's public key to encrypt the session key so that the encryption and decryption times are quicker.

The Federal Data Encryption Standard (DES) is one available form of single key encryption system.

No data security system is impenetrable. In any data security system, one must question whether the information protected is more valuable to an attacker than the cost of the attack. Public key encryption systems are most vulnerable if the public keys are tampered with.

An example will illustrate the problem. Suppose an originator wishes to send a private message to a recipient. The originator could download the recipient's public key certificate from an electronic bulletin board system and then encrypt a letter to the recipient with that public key and send it to him using an Internet E-mail message. Unfortunately, in the example, an interloper has generated a public key of his own with the recipient's user ID attached to it and substituted the phony public key in place of the recipient's real public key. If the originator unwittingly used the phony public key belonging to the interloper instead of to the intended recipient, everything would look normal because the phony key has the recipient's user ID. Now the interloper is in a position to decipher the message intended for the recipient because the interloper has the related private key. The interloper may even go so far as to reencrypt the deciphered message with the recipient's real public key and send it on to the recipient so that no one suspects any wrongdoing. Worse yet, the interloper can make apparently good signatures on behalf of the recipient using the phony private key because everyone will believe the phony public key is authentic and will utilize it to check the recipient's signatures.

To prevent this from happening, requires preventing anyone from tampering with public keys. If one obtained the recipient's public key reliably directly from the recipient, there is no doubt about the authenticity of the public key. However, where the public key is acquired from a source of uncertain reliability, there may still be a problem. One way to obtain the recipient's public key would be to obtain it reliably from a trusted third party who knows he has a good copy of the recipient's public key. A trusted third party could sign the recipient's public key, utilizing the trusted third party's private key, thus vouching for the integrity of the recipient's public key. However, to be sure that the third party's public key is authentic, requires that the sender have a known good copy of the third party's public key with which to check its signature. A widely trusted third party could specialize in providing a service of vouching for the public keys of other parties. This trusted third party could be regarded as a key server or as a certifying authority. Any public key certificates bearing the certifying authority's signature would be trusted as truly belonging to whom they appear to belong to. Users who desire to participate would need a known authentic copy of the certifying authority's public key so that the certifying authority's signatures could be verified.

Public key encryption systems are also subject to a vulnerability involving the use of bogus time stamps. A user may alter the date and time setting of the user's systems clock and generate either public key certificates or signatures that appear to have been created at a different time. He can make it appear that a document was signed earlier or later than it was actually signed or that the public's secret key pair was created earlier or later. This may have some type of benefit, for example, by creating circumstances which might allow him to repudiate a signature. In situations where it is critical that a signature have the actual correct date and time, an electronic equivalent of a notary can be utilized. An electronic notary would apply the notary's electronic signature to other people's electronic signatures, thus witnessing the date and time of the signed document. A notary could actually maintain a log of detached signature certificates and make it available for public access. The notary's signature would have a trusted time stamp which might carry more credibility than a time stamp on the original signature alone.

In most open network architectures, security is an ad hoc thing. Individual stations having access to the network may or may not choose to utilize encryption in their transmissions. If they do so, they alone are responsible for ensuring that they have authentic public keys of the persons with whom they are communicating. Some efforts have been made to standardize security procedures for such a network. For example, the current state of the development for secure

systems across the Internet is found in the Network Working Group Request For Comments No. 1421, dated February 1993 (RFC 1421). This document addresses proposals for privacy enhancement for Internet electronic mail, namely, message encryption and authentication procedures. That document is incorporated in its entirety by reference into this application.

A second proposal, Network Working Group Request For Comments No. 1422, also dated February 1993, addresses privacy enhancement for Internet electronic mail and particularly addresses certificate-based key management. This document is also incorporated by reference into this application in its entirety.

These proposals incorporate concepts utilized in the X.400 Message Handling System model of CCITT Recommendation X.400, the directory system Recommendation X.500 and the CCITT 1988 Recommendation X.509 directed to an authentication framework.

One of the problems with the prior art proposals is that they are directed primarily to Internet mail and do not cover a variety of the other types of services which might be performed over an open network. Specifically, they do not address secure transactions utilizing HTTP (Hypertext Transfer Protocol) and they do not address program-to-program communications.

Another problem with the prior art identified above is that for the most part these represent recommendations and proposals and do not represent actual implementations of systems for carrying out secure transactions.

Another problem with the prior art is that it does not provide a consistent application programming interface usable in all types of environments where secured transactions are needed.

Another problem with the prior art identified above is that it is not functionally complete and consistent, since it lacks specifications of certain types of control messages and protocols which are essential for correct functioning of certificate infrastructure.

Another problem with the prior art is that there is no consistent public key infrastructure which can actually and automatically provide the certifications required for a public key system.

Another of the problems with the prior art is that there is no hierarchical arrangement of certifying authorities which can cross policy certifying authority boundaries in pursuit of a global authorization system which will permit secure transactions to be undertaken worldwide transparently.

Another problem of the prior art is that there is no way for permitting secure transactions to cross organizational boundaries in a way that is convenient and transparent.

DISCLOSURE OF THE INVENTION

One advantage provided by the invention is that of providing a full, correct, consistent and very general security infrastructure which will support global secure electronic transactions across organizational, political and policy certifying authority boundaries.

Another advantage of the invention lies in providing consistent application programming interfaces which can be utilized in all types of electronic transactions for ensuring security and authenticity of all kinds of electronic documents.

Another advantage of the invention resides in the ability to provide efficient key management and distribution in a secure manner by several different ways, more effective than existing models, and in a manner which protects public keys from tampering.

Another advantage of the invention is the provision of trusted third party and notary services.

Another advantage of the invention is the provision of privacy and authenticity in the transmission of information by way of a general set of computer communication protocols and applications with consistent and easy to use interfaces to these functions.

Another advantage of the invention is the provision of a certificate-based public key system in which certificates are verifiable and readily available.

Another advantage of the invention is to provide a system where certificates are readily accessible and verifiable.

These and other advantages of the invention are achieved by providing a multi-hierarchical certification system for issuing and authenticating public keys used for all types of electronic transactions and applications.

Such a system may or may not comprise a distinguished certification authority representing a root node (or registration authority (RA) level) of a certification hierarchy. This certification authorities certifies one or more second certification authority at a policy certification authority (PCA) level. One or more third certification authorities are certified by each certification authority at a hierarchy certification authority (CA) level. Certification authority processes lower in the hierarchy than a certification authority process operating at the policy certification authority level all operate in according with the security policies set by a policy certification authority. Certification authorities, operating at the hierarchy certification authority level may certify other CA level computer processes in hierarchical fashion. Generally, one or more end users are certified by the lowest CA and form an end user level.

Multi-hierarchical certification system may be established as the number of autonomous certification hierarchies, operating without a single, top-level certification authority. In that case, some form of cross-certification is needed for their secure cooperation. Certification authorities at lower levels in the same or in different hierarchies may also cross-certify each other.

Each certification authority process in the hierarchy, except the RA process, holds a data structure electronically signed by at least one higher level computer process. In this manner, the certification authority processes are arranged in a certification hierarchy.

One or more of the certification authorities in the hierarchy may function as a trusted third party, as an electronic notary or as a common public key certificate repository.

A common certificate repository may contain public key certificates for all certification authorities in the hierarchy and/or certificate revocation lists for a plurality of all users or computer processes in the hierarchy.

Each user or certification authority of the infrastructure has access to a computer process which comprises appropriate certification software and storage areas for storing data structures known as public key certificates, for storing certificate revocation lists, and optionally for storing network map information, error code and message information and registration information.

Each computer process utilizes a common application programming interface for access to encryption and certification services. The application programming interface is a set of certification functions which can be invoked by commands or by messages, such as an http command, an email message or program to program communication.

The invention is directed to a certification system for issuance, distribution and verification of public key certificates which may be used for secure and authentic electronic transactions over open networks, which system includes computer processes implementing certification servers, certification clients and certification protocols, in which one or more first computer processes are associated with at least one initial (root) registration authority, one or more second computer processes are associated with policy certification authorities, one or more third computer processes are associated with certification authorities, and one or more end-user computer processes or application computer processes are associated with respective end-users or user applications. The second computer processes hold a data structure certified by said registration authority, the third computer processes hold a data structure certified either by one of said policy certification authorities or other certification authorities, and end-user or application computer processes hold a data structure certified by one or more of said certification authorities. As a result, users and applications of said system are logically located at end-points of certification chains in a certification infrastructure.

One or more of the computers in the infrastructure may function as a trusted third party, as an escrow agency, as an electronic clearing house for or insurer of electronic transactions, as an electronic Notary or as a common public key certificate repository.

A common certificate repository may contain public key certificates for all computers in the infrastructure and/or certificate revocation lists for a plurality of computers in the infrastructure.

Each computer of the infrastructure comprises storage areas for storing data structures such as electronic addresses, electronic identities or public key certificates, and for storing certificate revocation lists, for storing network configuration information, error code and messages and/or entity identification information.

Each computer utilizes a common application programming interface either for remote access to that process or for access to encryption, certification and other local services. The application programming interface comprises a set of primitives which can be invoked by commands, by messages, by remote procedure calls or by any other type of computer procedure invocations such as http commands or program to program communications.

The invention is also directed to a method of requesting and issuing a public key certificate in a certification system for secure communications containing computer processes arranged in a certification infrastructure, by generating a data structure containing the data items required for a public key certificate at a requesting computer process, including a public key, self-signing the data structure and sending the signed data structure as a certificate signature request to a computer process authorized as an issuing certification authority, and verifying the authenticity of said request at a computer process authorized as an issuing certification authority, and if authentic, certifying and returning the data structure in a certificate signature reply. The received signed certificate, or a copy, is stored either at said requesting computer process or at a common certificate repository. This method is invoked when adding a new entity to a certification infrastructure or upon expiration of an existing certificate.

The invention is also directed to a method of verifying a signed data structure sent from a sender to a receiver by obtaining a public key certificate for every computer in the infrastructure between the sender and a common point of trust in the infrastructure and verifying the authenticity of each signature iteratively, beginning with the common point of trust. Public key certificates for every computer in the infrastructure between the sender and a common point of trust may be obtained from a common repository or from respective individual computers. To ensure validity of a certificate, is verified against one or more or preferably all relevant certificate revocation lists and/or by a common repository. A public key certificate of a sender may also be verified by a direct inquiry to the certification authority which issued that certificate.

The invention is also directed to a method of validating public key certificates by using the certificate revocation lists of each computer process between a computer process or user whose certificate is being validated and a point of trust in common with the computer process or user which is validating the certificate to ensure the certificates being used in the validation process do not appear on any certificate revocation list.

The invention is also directed to a method of updating certificates by:

a. at a first computer process, which possesses a certificates to be updated, updating the current certificate by
  a.1. receiving a new signed certificate from a computer process which is authorized to issue the new signed certificate,
  a.2. revoking the current certificate previously used for verification of certificates of subordinate computer processes,
  a.3. issuing new certificates to all subordinate computer processes for which certificates had been previously signed by the first computer process and copying to all subordinate computer processes the new certificate to be used for verification of new subordinate certificates, and
b. iteratively performing the distribution of the new certificate to all subsequent subordinate computer processes, until all computer processes subordinate in the infrastructure to said first computer process have the new certificates.

The invention is also directed to a method of adding a new computer process to the infrastructure by adding a new component to a representation of a certification infrastructure at a location indicative of where the said computer process is to be added, creating entries in a certificate storage database at least at both said new computer process and at the computer process authorized to certify the said new process, and obtaining a signed certificate for the said new computer process from said computer process authorized to certify the new process and storing it at the said new computer process.

The invention is also directed to a method of deleting an existing computer process from the infrastructure by notifying at least all computer processes certified by the existing process being deleted that said existing computer process is being deleted, revoking all certificates signed by said first computer process at said computer processes certified by the existing process being deleted, if any; and obtaining new certificates for each computer process previously being certified by the said existing computer process being deleted from another certification authority being authorized to certify these computer processes in the new certification infrastructure. All certificates revoked are added to a certificate revocation list.

The invention is also directed to a method of restructuring at least part of the certification infrastructure by deleting one

or more certification authorities and adding said one or more certification authorities or new certification authorities so as to derive a modified form of the certification infrastructure.

Still other objects and advantages of the present invention will become readily apparent to those skilled in this art from the following detailed description, wherein only the preferred embodiment of the invention is shown and described, simply by way of illustration of the best mode contemplated of carrying out the invention. As will be realized, the invention is capable of other and different embodiments, and its several details are capable of modifications in various obvious respects, all without departing from the invention. Accordingly, the drawing and description are to be regarded as illustrative in nature, and not as restrictive.

BRIEF DESCRIPTION OF DRAWINGS

FIG. **27** is a flow chart of a Certificate_Verify process.

FIG. **28** is a flow chart of a Main Certification Server process.

BEST MODE FOR CARRYING OUT THE INVENTION

FIG. **1A** is a logical representation of a hierarchical security or public key infrastructure in accordance with the invention. Each block illustrated in FIG. **1** represents a certification authority which either uses or performs function within the public key infrastructure, or both. Although in actuality, each of the blocks in FIG. **1** is connected to a communications network so that each certification authority may exchange information with any other, a logical hierarchical arrangement is shown with the various levels representing where a particular certification authority is positioned in the certification hierarchy. Certification may be as simple as merely signing a public key certificate of a "subordinate" user, certification authority or computer or it may involve carrying out a full set of activities specified by a security policy. At the highest level of the certification hierarchy may be the root of the hierarchy, a Policy Registration Authority (PRA), with global jurisdiction. This PRA is equivalent to that envisioned for an Internet policy registration authority in RFC **1422**. Beneath the policy registration authority are Policy Certification Authorities (PCA), each of which defines a particular set of certification policies which differ from PCA to PCA. Policy certification authorities set the standards for their particular certification sub-hierarchies. A policy certification authority could, for example, be a standards body of a particular national government. Alternatively, a policy certification authority might be the chief information officer of a multinational corporation. What is important is that organizational entities operating under a substantively different set of policies should interface through their policy certification authorities. Below the policy certification authorities are certification authorities such as **120**, all of which follow the policies set by PCA **110**. Certification authorities can then certify sub-certification authorities in a hierarchical fashion until ultimately the end users are certified at the bottom of the hierarchy.

In FIG. **1**, as an example, policy certification authority **110** may be established as a national certification authority, say, for example, for the U.S.A. Underneath the policy certification authority are certification authorities **120** which could, pursuing the hypothetical, be established for each state in the United States. Beneath that could be certification authorities **130** for county governments, and under that certification authority for cities at **140** and ultimately down to the residential user level at **150**. The particular division and assignment of certification authorities are established by the policies established by the PCA. Policy certification authority **115** might service a number of corporations each having their own certification authorities **125**. Company wide CA **125** might then certify a number of operations such as **135** within the company. Each operation might then certify its divisions **145** and the divisions might certify departments **155** and the departments might certify working groups **165** and user's **166**. The working groups might then certify site **175** and user **176** and the site might certify, ultimately, end organizational users **186**.

Each of the blocks in FIG. **1A** is implemented as a computer process running on a computer. Depending on implementation, several certifications may be implemented at the same computer. More than one block, e.g. more than

one certification authority, may run on a single computer. The particular kind of computer is not particularly important although multitasking Unix machines are preferred, such as those made by Sun, Hewlett-Packard, etc. In addition to the usual compliment of input output devices and system software, each computer is equipped with a network access permitting it to communicate over a network with other stations. Each computer has a memory and storage capabilities in quantities and type which vary from machine to machine.

FIG. **1B** is similar to the system shown in FIG. **1A** except there is no single root authority. Certification authority is distributed across the network in a matrix rather than hierarchical fashion. With this architecture, certifying authorities analogous to policy certifying authorities cross certify each other, so that there is common agreement across the network as to who is authorized to certify.

FIG. **2** illustrates an allocation of memory or storage or both to certain types of data base functions. A registration data base for certification authorities and users exists at **210**. A network map and certification infrastructure data base is shown at **220**. A certificate storage data base is shown at **230**. A certification revocation list (CRL) data base is shown at **240** and an error code/message data base is shown at **250**. Access to a data base may be through a data base management system, typically and preferred, and the various data bases may be maintained as separate data bases or as components of one large data base. The data base functionality is important and not particularly where, nor in what manner of storage the records for the data base are maintained. Typically, the allocation between memory and longer term storage is made on the basis of performance characteristics needed.

In accordance with the invention, secure electronic documents and the handling of public keys in an open network, such as Internet, are based on some type of certificate. A certificate is specially constructed data structure which contains the user's public key. Further, a certificate contains unique identification of the public key owner and some additional parameters related to the validity of the certificate. In order to guarantee the integrity, authorization and originality of certificate data, each certificate must be issued by an authority, in this context, called a Certification Authority (CA). The Certification Authority vouches for the identity of the public key owner, for the integrity of the public key itself, for the binding between the public key and the owner's identity, and optionally for some additional capabilities of the certificate owner in the electronic environment. This guarantee is reflected in the certificate through the identity of the authority, together with the authority's digital signature to the certificate. Certificates may further may contain references to the types and purposes of public keys, to the relevant certification policies and eventually to the authorization privileges of certificate owners. Certificates may contain other parameters relevant for the purposes and usages of certificates and public keys.

A certificate is a data structure. A sample of such a data structure is represented in FIG. **3**. The version number of the certificate, shown at item **300**, is intended to facilitate orderly changes in certificate formats over time. Typically, version numbers may be those utilized in the X.509 recommendation by default.

Serial Number **310**, is a short form, unique identifier for each certification generated by an issuer. A serial number is unique only to an issuer. That is, an issuer will not issue two certificates with the same serial number. The serial number is used in certificate revocation lists (CRL's) to identify revoked certificates.

Item **320** represents the electronic signature of the issuer together with the algorithm and parameters utilized to sign the certificate.

Item **330** represents the issuer's name which is a representation of the issuer's identity, preferably in the format of a distinguished name as set forth in the X.500 directory system.

The validity period is a pair of date and time indications indicating the start and end of the time period over which a certificate is valid.

The subject name, shown at item **350**, is also a distinguished name such as that utilized in an X.500 directory system.

Item **360** shows the public key of the subject which is being certified by the certifying authority.

Finally, item **370** contains additional information which is optional which might be useful to the purposes discussed above.

The registration process for a certification authority which desires to participate in the security or public key infrastructure begins with an application which provides the various kinds of information required by the policy certifying authority. The information on the application is verified either automatically or manually, depending on the policy, and if the application meets acceptance criteria, the certification process may begin.

Certification begins with a message sent from the station desiring certification to the certifying authority or by receiving that notification in any other way. Typically, this is done in a Certificate_Signature_Request message. The format of the Certificate_Signature_Request includes a certificate filled in with at least the public key which the requesting entity desires to have certified. The submission may be self-signed using the requestor's private key and transmitted to the CA for signature. It is possible, of course, to include all of the application information as part of the Certificate_Signature_Request sent to the CA with the main purpose to allow the receiving CA to verify the correctness and authorization of the received request. When the CA receives the Certificate_Signature_Request, the information contained therein is validated in accordance with the policies established by the PCA, and if the information is correct, the certifying authority issues a Certificate Signature_Reply message returning to the requesting entity a signed certificate. When the requesting entity receives the Certificate_Signature_Reply message, it undertakes a Receive_Certificate process which verifies the signature on the certificate and stores it in a local certificate data base after verifying that the public key contained in the certificate corresponds to the entity's private key.

As used herein, a descriptor such as Certificate_Signature_Request can refer to either a process which generates a Certificate_Signature_Request message, the message itself, a command or any other method which initiates the certification process. These distinct usages are apparent from the context.

Once the initiator is in possession of a signed certificate from a certifying authority, the entity is prepared to engage in other secured transactions as described hereinafter. If the entity is a CA it may perform other certification functions. If the entity is end user, it may perform secure transactions and certification functions.

FIG. **4** illustrates how the public key infrastructure in accordance with the invention can be utilized to verify

transactions. In this example, assume that user U2 (**430**) sends a signed message to user U1 (**450**). It is convenient and preferred for each user, such as U1, to have certificates stored in their certificate storage data base **230**, for themselves and for each station between the user U1, and the policy registration authority.

Although user U2 could have sent a certificate with a signed message, for this example, we will assume that U2 did not include a certificate. Thus, for user U1 to have confidence that the signed message is genuine and that it originated with U2, the signature must be verified. To do this, user U1 sends a Certificate_Request message to user U2 and to certifying authorities CA2 and CA3. Since CA1 is in the direct chain of hierarchy between U1 and the PRA, the certificate storage data base of station U1 presumably contains a certificate for CA1. User U1 sends a Certificate_Request Message to user U2, CA2 and CA3. When user U1 receives Certificate_Reply messages from these entities, their certificates are extracted, verified and stored in the certificate storage data base. They are then utilized as follows:

Since the certificate received from station CA2 is signed by CA1 and since U1 already has a certificate of CA1 in the certificate storage data base, CA2's certificate can be authenticated by using the locally stored version of CA1's public key. If it verified properly, then CA2's certificate is accepted as valid. Since CA3 was certified by CA2 and since U1 now has a valid certificate for CA2, which it placed in storage when received, U1 can verify the certificate of CA3 by utilizing the public key for CA2 to verify the signature of the certificate of CA3. If it verifies properly, then the certificate for CA3 is accepted as valid and one can utilize the public key contained therein to verify the certificate of station U2 by verifying U2's certificate signature with the public key contained in CA3's certificate. Thus, having a known valid certificate for U2, U1 may verify the signed message using the public key of U2's and thus have considerable confidence that the message is authentic and that no public keys have been tampered with. Station CA1 represents the "common point of trust" in the hierarchy in that it is the lowest point in the hierarchy which is common to both the sending and receiving stations.

FIG. **5** illustrates the process by which a signature may be verified.

Once U1 has determined that he has a valid public key for user U2 using the verified and validated certificates, there are two ways ensuring that the signature is authentic. These two ways relate to how the signature was generated. As discussed above, in one signature mode, the entire document is encrypted with the private key of the sender. Thus, if one decrypts the encrypted contents using a public key (**510**) if clear text or some other recognizable message results (**520**), the signature is authentic (**560**). On the other hand, in a second signature mode a contents digest is utilized to sign the document. One would decrypt the encrypted contents digest using PK_{U2} (**530**), calculate the digest of the contents independently using the clear text contents (**540**) and if the decrypted contents digest is identical with the calculated contents digest (**530**), the signature is authentic (**560**).

There are three primary reasons for revocation of a certificate. The first is the owner suspected compromise of a private key. The second is a change of user or CA affiliation. This third is certificate expiration.

As discussed above, to validate a certificate reliably, the validator must ensure that none of the certificates utilized in validation has been revoked. To ensure that, the validator must have a correct certification revocation list from the common point of trust to the entity whose certificate is being validated. As shown in FIG. 6, a certificate revocation list is a data structure which contains a signature of the issuing party (**600**) together with algorithm ID and parameters used to sign the list, the electronic ID of the issuer (**610**), the last update date and time (**620**), the next scheduled update date and time (**630**) and a list of revoked certificates (**640**), arranged as shown, for example, in block **650**. Revoked certificates are denoted by their sequence numbers in a sequential order and for each sequence number list the serial number of the certificate being revoked and the date and time of its revocation.

To retrieve the current CRL of all relevant CA's, a user can send a CRL_Request message to the station and receive the list back in the form of a CRL_Reply message. When the list returns, it may be stored in the CRL data base using the CRL_Store command. In some systems, it may be preferred to use a common repository which maintains authenticated copies of CRL's for all CA's in the entire system. A CRL may then be obtained by CRL Request message directed to the common repository and receive the response back via CRL_Reply message from the common repository. When using a common repository, a CA may send a copy of its current CRL to the common repository using the CRL_Store message. Once it has been successfully received by the common repository, a reply will be sent to the sending CA using the CRL_Confirm message format. As discussed in conjunction with the verification process of FIG. 4, current CRL's are required in order to properly authenticate and verify the certificates.

When using the public key infrastructure of the invention, it is often required to fetch certificates. These may be fetched in advance or on an as needed basis. They may be fetched from owner's, issuers or certificate repositories. They are fetched using a Certificate_Request message listing the identification of the entity whose certificate is needed and the certificate is returned using a Certificate_Reply message. Certificates can be fetched by program-to-program communications, interactive HTTP, store and forward mail, or any type of communications.

The Verify_Certificate process can be utilized two ways. First, it can be utilized to verify all certificates between the entity for which a certificate is being verified to the common point of trust with the verifier. This will also be based on usage of CRL's to ensure that the certificate certified and all other certificates used in the process are still valid. The second option utilizes direct verification by sending a Verify_Certificate message to a common repository which is known to be trusted and the common repository responds with a currently valid certificate of the entity being validated. In this mode, no CRL's are needed.

FIGS. 7–27 describe a set of processes which collectively form the certification system, functions, and certification infrastructure of this invention. The processes may be invoked singularly or in combination and may be called by any other process. The commands and processes described in FIGS. 7–24 thus present a set of protocol and programming primitives which may be invoked either directly by a user or by part of an application process running on the user's or CA's computer. There is thus a standardized interface for all security functions desired by any computer system or user.

A flow chart of the registration process for users and CAs is shown in FIG. 7. The process starts at **700** and a new user/CA sends (**705**) the application for registration to the policy certifying authority (PCA).

The PCA investigates the requester and the facts in the application in accordance with the PCA's policy (**710**). If disapproved, a reject message is sent (**720**) whereas if approved (**715**) the approval and instructions are sent to the Applicant (**725**), a new entity is added to the registration data base and the Add_New_CA/User process is performed. If Applicant has not already acquired the software, the Applicant acquires PKI software and installs it on his system (**730**). After registration, using the software, the Applicant performs the Certificate_Request process, (discussed hereinafter **735**), self signs the certificate and sends it to the certification authority. If the certificate fails certain policy or format checks (**740**-N), a Certificate Signature_Reject message is prepared and sent to the Applicant. The Applicant may then again modify the request and submit it as previously indicated at block **735**. If the Certificate_Signature_Request is accepted (**740**-Y), the CA verifies the authenticity of the request, signs the certificate and performs Certificate_Signature Reply (**750**). When the Applicant receives the certificate contained in the Certificate_Signature_Reply message, Applicant performs the Receive_Certificate process (**755**) and the certification process is complete.

The Certificate_Signature_Request process is described in FIG. **8**. The process begins at **800** and the Applicant generates a certificate (**810**) including a public key. The certificate is filled in to the extent possible, absent, of course, the signature of the certifying authority. At **820**, the Applicant adds whatever other information may be required by the PCA policy and formats the certificate into a request format. The Applicant self signs the certificate (**830**) and sends the self signed request to the CA for signing (**840**) using the Certificate_Signature_Request message format. The process ends at **850**.

FIG. **9** is a flow chart of the Certificate_Signature_Reply process. The process starts at **900** by receiving a Certificate_Signature_Request message (**905**). The receiving authority authenticates the request in accordance with the policies set down by the policy certifying authority (**910**). The request message format is checked for compliance with certain formatting criteria (**915**). If it fails, a Certificate_Signature_Reject message is sent (**920**). If it passes, a check is made to see if this involves a new entity (**925**). If it does not (**925**-N), the certifying authority signs the certificate (**930**), marks the old certificate revoked including a date time stamp (**935**), and adds the old certificate to the certificate revocation list (**940**). If the request comes from a new entity (**925**-Y), the new certificate is signed (**950**). The signed certificate is stored in a certificate storage data base and/or forwarded to a common certificate repository (**955**). The signed certificate is sent to the requester in a Certificate_Signature_Reply message (**960**) and the process ends.

FIG. **10** is a flow chart of the Receive_Certificate process. The process begins at item **1000** and, when a Certificate_Signature_Reply or Certificate_Resign_Reply message is received (**1010**), the message is authenticated (**1020**). The public key contained in the signed certificate is compared with the public key corresponding to the private key used to sign the Certificate_Signature_Request (**1030**). If the keys agree, the signed certificate from the incoming message is stored in the certificate storage data base (**1040**) and the process ends.

FIG. **11** is a flow chart of a Certificate_Signature_Reject process. The process begins at (**1100**) upon receipt of a Certificate_Signature_Request error code (**1110**). The error message associated with the error code is retrieved from the error message data base (1120) and the error code and error message are sent together as part of a Certificate_Signature_Reject message to the requesting entity (1130) and the process ends. FIG. 12 is a flow chart of a Certify_CA or Certify_User process. The process begins (1200) when a Certify_CA or Certify_User command is received (1205) from a local user/CA administrator. New keys for the entity being certified are generated (1210) and the Certificate_Signature_Request process is executed (1215). While a request process is outstanding, all security functions are disabled except for Certificate_Signature_Reply (1220) until either a Certificate_Signature_Reject message (1225) or a Certificate_Signature_Reply message is received (1230). If a Certificate_Signature_Reject message is received (1225-Y), another attempt is made to submit a certificate for certification using the Certificate_Signature_Request process. Of course, a counter may be utilized to limit the number of times this loop is traversed. If a Certificate_Signature_Reply message is received, security functions are enabled (1230) and the newly received certificate is processed in accordance with the Receive_Certificate process (1235). If the entity being certified is a CA, all subordinate units must be updated with the new certificate by performing a Certificate_Path_Update function (1240) and the process ends.

FIG. 13 is a flow chart of an Update_CA process. The process begins at 1300 where a check is made to see if the CA certificate expiration date is greater than today's date (1310). If it is not, (1310-N) a certain interval of time will expire (1320) prior to rechecking the expiration date. Once CA certificate expiration date exceeds today's date (1310-Y) the process calls Certify_CA (1330) and the process ends.

FIG. 14 is a flow chart of a Certificate_Resign Request process. The process begins at 1400 and a Certificate_Resign_Request command is received from the local user/CA administrator (1410). A new key pair is generated (1420) and used for generating a new certificate for the entity (1430). The other information required by the PCA policies are incorporated into a request message (1440) and the new certificate is signed by the local entity using the private key corresponding to the old certificate (1450). The signed Certificate Resign_Request message is sent then to the CA for signing (1460) and the process ends.

FIG. 15 is a flow chart of a Certificate_Resign Reply message. The process begins at 1500 and a Certificate_Resign_Reply message is received (1510). When that message is received, the Receive_Certificate process is executed (1520) and if the entity receiving the message is a CA, the Certificate_Path_Update process is executed to notify all subordinate entities of the new certificate (1530) and the process ends.

FIG. 16 is a flow chart of a Certificate_Path_Update process. The process begins (1600) and the local entity identifies all subordinate CA's or users, if any (1610). If there are some, the entity is a CA and the CA will issue new certificates to each subordinate CA and user using a Certificate_Resign_Reply message (1620) and the process ends.

FIG. 17 is a flow chart of an Add_New_CA process.

The process begins (1700) and a new CA is added to the network map certification infrastructure data base maintained by the PCA at the location specified by the registration data base entry (1710). An entry is also created in the network map and certification infrastructure data base at the superior CA specified in the registration data base (1720). The new entity performs Certify_CA or Certify_User (1730) and the process ends.

FIG. 18 is a flow chart of a Delete_CA process. The process begins (1800) by specifying the CA to be deleted (1805). All subordinate entities of the CA to be deleted are identified (1810) and a Delete_CA message is sent to all subordinate CA's specifying the identification of the CA being deleted (1815). At each CA, all certificates issued by the Deleted_CA are revoked and added to the certificate revocation list (1820). A determination is made whether or not all subordinate units are to be removed (1825). If they are not, (1825-N) the Attach_Subordinates process is executed directed to a selected CA, preferably the next higher CA (1845) and the process ends. If they are to be removed, a check is made to determine whether the subordinates are to be attached to another CA (1830). If they are, (1830-Y) the Attach_Subordinates process is executed directed to the CA where attachment is desired (1835) and the process ends. If attachment to another CA is not desired (1830-N), if a subordinate unit is a CA, this process (the Delete_CA process) is performed recursively for all subordinate CAs and the process ends.

FIG. 19 is a flow chart of the Attach_Subordinates process.

The process starts at (1900) by identifying all CA's or users immediately below a CA being deleted (1910). The CA immediately above the CA being deleted is also identified (1920). For each immediately subordinate CA or user, the Certificate_Signature_Request process must be performed directed to the desired CA followed by process Receive_Certificate (1930) process. If the subordinate entity is a CA, once it is attached to the new desired CA, it must do a Certificate_Path_Update to update all of its subordinate units (1940) and the process ends.

FIG. 20 is a flow chart of a Revoke_Certificate process. The process starts at 2000 and the certificate to be revoked is identified (2010). The certificate identified for revocation is deleted from the certificate storage data base (2020) and the information from the certificate relevant to a CRL is stored using the process CRL_Store (2030) and the process ends.

FIG. 21 is a flow chart of the CRL_Store process. The process begins (2100) and the certificate to be revoked is identified (2110). The information required for entry in the CRL data base locally is extracted from the certificate identified (2120) and a record for the revoked certificate is added to the CRL. If a common repository is in use, the entity sends a CRL_Store message to the common repository (2130). If a CRL Confirm message is received back from the common repository (2140) the process ends. Otherwise, after a period of time, another attempt will be made to update the CRL at the common repository (2150). Of course, a counter can be utilized to limit the number of times the loop is traversed before a failure is declared.

FIG. 22 is a flow chart of a CRL_Confirm process. The process starts (2200) and incoming messages are monitored (2210). If a CRL_Store message is received (2220-Y) the CRL information is extracted from the message and stored in the common CRL data base (2230). Once the storage in the common CRL data base is confirmed, a CRL_Confirm message will be sent back to the CRL_Store message sender (2240) and message monitoring will resume. If a CRL_Store message is not received, (2220-N) monitoring of messages will resume.

FIG. 23 is a flow chart of the CRL_Request process. The process begins (2300) and a CRL_Request command with a list of CA's or a CRL_Request message with a list of CA's arrives (2305). For each CA on the list, (2310) a determination is made whether the CA is the local station or whether the message is addressed to a common repository which is, in fact, this station (**2315**). If the command or request is directed to this station, the station will access a local data base and retrieve the CRL's requested (**2320**) and package them for return to the requester (**2325**). If the command or message is not directed to this station (**2315**-N) a check is made to determine whether or not a common repository is in use in the system (**2330**). If it is, the CRL request message will be sent to the common repository along with the list of ID's of the CA's whose CRL's are needed (**2335**). If all CRL's are obtained by this method, (**2340**) the process ends. If it is not, (**2340**-N) or if a common repository is not in use (**2330**-N), a CRL_Request message will be sent to each CA on the list for which a CRL has not been received (**2345**). After the messages are sent, incoming messages will be monitored (**2350**) to determine whether a CRL_Reply message has been received (**2355**). If it has not been, monitoring of incoming messages will resume. If it is received, the CRL's included in the CRL_Reply message will be extracted and packaged for return to the requesting process (**2360**) and the process ends.

FIG. **24** is a flow chart of the CRL_Reply process. The process begins (**2400**) and, when a CRL_Reply message or return from a CRL_Request command is received (**2410**), the return CRL is stored in the CRL data base using the CA identification as a key (**2420**) and the process ends.

FIG. **25** is a flow chart of a Certificate_Request process. The process begins (**2500**) and when a Certificate_Request command is received containing the ID of a CA or user (**2510**), a Certificate_Request message is sent to the user or CA (**2520**). Messages are monitored (**2530**) until a Certificate_Reply message is received (**2540**). Once it is received, the certificate is extracted from the message, verified and stored in the local certificate data base (**2550**) and the process ends. If a Certificate_Reply message is not received, monitoring will continue until a time out is exceeded in which case the process fails.

FIG. **26** is a flow chart of a Certificate_Reply process. The process begins (**2600**) and a Certificate Request message is received (**2610**). The ID of the station whose certificate is requested is extracted from the message (**2620**) and the local certificate data base is accessed using the certificate serial number to retrieve the requested certificate (**2630**). The requested certificate is inserted into a Certificate_Reply message and sent to the requester (**2640**) and the process ends.

FIG. **27** is a flow chart of the Certificate_Verify process. The process begins (**2700**) and when a Certificate_Verify message containing a certificate is received (**2705**), the certificate is extracted. The certificate is verified and if successful, stored in the local data base. If verification fails, the error message is returned to the issuing entity.

The certification functions and protocols described in FIGS. **7–27** constitute a set of relatively independent subroutines which generally can be invoked by a direct command from a local process, IO device or received message. FIG. **28** illustrates how particular processes are invoked by these methods. The process is essentially a certification server process which begins (**2800**) and continuously monitors commands (**2810**) and incoming messages (**2820**). If an incoming message is received at the station, the type of message is determined (**2830**) and the appropriate process invoked based on the type of message (**2840**). Incoming commands are also monitored (**2810**). The appropriate process can be invoked manually by commands as well. On an

ongoing basis, therefore, locally generated commands and incoming messages are both monitored continuously and the appropriate process started, in response to either, to handle the command or message.

In the manner described, an entire public key infrastructure can be created and the components of the infrastructure interrelated so as to handle certificates in an automated and convenient manner with a consistent certification functions which are easy to use directly or as part of a program. The techniques made available by the infrastructure can thus be applied in entire universive electronic transactions, beyond merely simple secure E-mail.

In addition, the problems and shortcomings of the prior art are eliminated using the disclosed public key infrastructure described herein.

In this disclosure, there is shown and described only the preferred embodiment of the invention, but, as aforementioned, it is to be understood that the invention is capable of use in various other combinations and environments and is capable of changes or modifications within the scope of the inventive concept as expressed herein.

In this disclosure, there is shown and described only the preferred embodiment of the invention, but, as aforementioned, it is to be understood that the invention is capable of use in various other combinations and environments and is capable of changes or modifications within the scope of the inventive concept as expressed herein.

What is claimed is:

1. A certification system for issuance, distribution and verification of public key certificates which may be used for secure and authentic electronic transactions over open networks, comprising computer processes implementing certification servers, certification clients and certification protocols, in which:

a. one or more first computer processes are associated with at least one initial (root) registration authority,

b. one or more second computer processes are associated with policy certification authorities,

c. one or more third computer processes are associated with certification authorities, and

d. one or more end-user computer processes or application computer processes are associated with respective end-users or user applications, and

e. said one or more second computer processes hold a data structure certified by said registration authority, said one or more third computer processes hold a data structure certified either by one of said policy certification authorities or other certification authorities, and end-user or application computer processes hold a data structure certified by one or more of said certification authorities,

whereby users and applications of said system are logically located at end-points of certification chains in a certification infrastructure.

2. The system of claim **1** in which some of said certification authorities may also function as Trusted Third Parties.

3. The system of claim **1** in which some of said certification authorities may also function as Escrow Agencies.

4. The system of claim **1** in which some of said certification authorities may also function as a clearing house for or insurer of electronic transactions.

5. The system of claim **1** in which some of said certification authorities may also function as Electronic Notaries.

6. The system of claim **1** in which some of said certification authorities may also function as common repositories for electronic identities and public key certificates (Directories).

7. The repository systems of claim 6 in which said common repositories may hold electronic identities and/or public key certificates of other certification authorities, users, applications and other components in the certification system.

8. The repository systems of claim 6 in which said common certificate repositories may also hold certificate revocation lists for a plurality of computer processes in the certification system.

9. The certification system of claim 1 in which one or more computer processes may access a storage area for storing various identification, authentication and authorization data structures, certificates and certificate revocation lists.

10. The system of claim 1 in which one or more computer processes of the certification system may access storage areas for storing or fetching network configuration information, error codes and messages, or entity identification information.

11. The certification system of claim 1 in which said data structures may be electronic addresses, electronic identities or public key certificates.

12. The certification system of claim 1 in which each computer process may utilize a common application programming interface (API) either for remote access to that process or for access to encryption, certification and other local services.

13. The certification system of claim 12 in which each computer process utilizes said common application programming interface comprising a set of programming primitives implementing certification protocol steps.

14. The set of programming primitives of claim 13 in which one or more members of the set can be invoked by commands, by messages, by remote procedure calls or by any other type of computer procedure invocations.

15. The API system of claim 12 in which the applications programming interfaces may be invoked by http commands.

16. The API system of claim 12 in which the applications programming interfaces may be invoked by E-mail messages.

17. The API system of claim 12 in which the applications programming interfaces may be invoked by a program to program communication.

18. In a certification system for secure communications containing computer processes arranged in a certification infrastructure, a method of requesting and issuing a public key certificate, comprising:

a. at a requesting computer process, generating a data structure containing the data items required for a public key certificate, including a public key, self-signing the data structure and sending the signed data structure as a certificate signature request to a computer process authorized as an issuing certification authority, and

b. at said computer process authorized as an issuing certification authority, verifying the authenticity of said request, and if authentic, certifying and returning the data structure in a certificate signature reply.

19. The method of claim 18, further comprising:

storing the received signed certificate at said requesting computer process.

20. The method of claim 18 further comprising:

storing the received signed certificate or copy of a signed certificate at a common certificate repository.

21. The method of claim 18 performed when adding a new entity to a certification infrastructure, which entity may be policy certification authority, certification authority, application or end-user.

22. The method of claim 18, performed upon expiration of an existing certificate, where the new certificate may contain either the existing or a new public key.

23. In a global network with secure communications containing computer processes arranged in a certification infrastructure, a method of verifying a signed data structure sent from a sender to a receiver, comprising:

a. obtaining a public key certificate for every computer process in the infrastructure between the sender and a common point of trust in the infrastructure and,

b. verifying the authenticity of signatures iteratively, beginning with the common point of trust.

24. The method of verifying of claim 23 in which a public key certificate for every computer process in the infrastructure between the sender and a common point of trust is also verified against all relevant certificate revocation lists.

25. The method of verifying of claim 23 in which a public key certificate of a sender may also be verified by a direct inquiry to the certification authority which issued that certificate.

26. The method of verifying of claim 23 in which a public key certificate for every computer process in the infrastructure between the sender and a common point of trust may be obtained from respective individual computer processes.

27. The method of verifying of claim 23 in which a public key certificate for every computer process in the infrastructure may also be obtained from a common repository.

28. In a certification system for secure communications containing computer processes arranged in a certification infrastructure, a method of validating public key certificates comprising:

using the certificate revocation lists of each computer process between a computer process or user whose certificate is being validated and a point of trust in common with the computer process or user which is validating the certificate to ensure the certificates being used in the validation process do not appear on any certificate revocation list.

29. The method of claim 28 in which retrieved certificate revocation lists are stored locally in the computer at which the certificate is being validated.

30. In a computer system for secure communications containing computer processes arranged in a certification infrastructure, a method of updating certificates comprising:

a. at a first computer process, which possesses a certificates to be updated, updating the current certificate by

a.1. receiving a new signed certificate from a computer process which is authorized to issue the new signed certificate,

a.2. revoking the current certificate previously used for verification of certificates of subordinate computer processes,

a.3. issuing new certificates to all subordinate computer processes for which certificates had been previously signed by the first computer process and copying to all subordinate computer processes the new certificate to be used for verification of new subordinate certificates, and

b. iteratively performing the distribution of the new certificate to all subsequent subordinate computer processes, until all computer processes subordinate in the infrastructure to said first computer process have the new certificates.

31. In a certification system for secure communications containing computer processes arranged in a certification infrastructure, a method of adding a new computer process to the infrastructure comprising:

a. adding a new component to a representation of a certification infrastructure at a location indicative of where the said computer process is to be added,

b. creating entries in a certificate storage database at least at both said new computer process and at the computer process authorized to certify the said new process,

c. obtaining a signed certificate for the said new computer process from said computer process authorized to certify the new process and storing it at the said new computer process.

32. In a certification system for secure communications containing computer processes arranged in a certification infrastructure, a method of deleting an existing computer process from the infrastructure comprising:

a. notifying at least all computer processes certified by the existing process being deleted that said existing computer process is being deleted,

b. revoking all certificates signed by said first computer process at said computer processes certified by the existing process being deleted, if any;

c. obtain new certificates for each computer process previously being certified by the said existing computer process being deleted from another certification authority being authorized to certify these computer processes in the new certification infrastructure.

33. The method of claim 32 further comprising:

adding all certificates revoked to a certificate revocation list.

34. In a certification system for secure communications containing computer processes arranged in a certification infrastructure, a method of restructuring at least part of the certification infrastructure by deleting one or more certification authorities and adding said one or more certification authorities or new certification authorities so as to derive a modified form of the certification infrastructure.

* * * * *

EXHIBIT B

US006826694B1

(12) United States Patent
Dutta et al.

(10) Patent No.: US 6,826,694 B1
(45) Date of Patent: Nov. 30, 2004

(54) HIGH RESOLUTION ACCESS CONTROL

(75) Inventors: **Partha P. Dutta**, San Jose, CA (US); **Mahesh M Kumar**, Fremont, CA (US); **Michah Lerner**, Lakewood, NJ (US)

(73) Assignee: **AT&T Corp.**, New York, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/422,952**

(22) Filed: **Oct. 22, 1999**

Related U.S. Application Data

(60) Provisional application No. 60/105,188, filed on Oct. 22, 1998.

(51) **Int. Cl.**^{7} **G06F 11/30**; G06F 15/173
(52) **U.S. Cl.** **713/201**; 709/224
(58) **Field of Search** 713/201, 200, 713/151, 152, 154; 709/224, 225, 229

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,473,607 A		12/1995	Hausman et al.	370/85.13
5,983,270 A	*	11/1999	Abraham et al.	709/224
6,219,706 B1	*	4/2001	Fan et al.	709/225
6,584,508 B1	*	6/2003	Epstein et al.	709/229

FOREIGN PATENT DOCUMENTS

EP	0 762 707 A2	8/1997
WO	96/05549	2/1996

OTHER PUBLICATIONS

Bellovin, S..M., "Network Firewalls", IEEE Communications Magazine, vol. 32, No. 9, Sep. 1, 1994, pp. 50–57, XP000476555; p. 52, col. 1, In. 60; p. 54, col. 2, In 30.

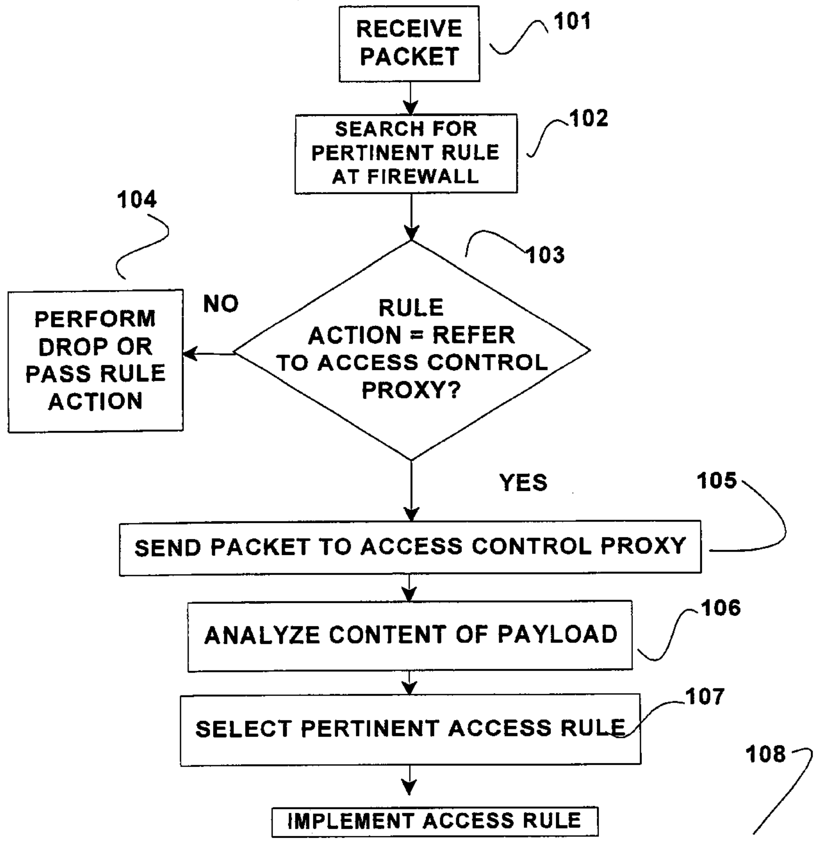
* cited by examiner

Primary Examiner—Matthew Smithers

(57) ABSTRACT

A system and method for high resolution access control for packetized information. A packet is received at a firewall. A rule corresponding to header information in the packet prescribes referring the packet to an access control proxy. The access control proxy analyzes the contents of the packet, and identifies a rule based upon the contents. The rule is implemented at the firewall.

1 Claim, 3 Drawing Sheets


RECEIVE PACKET
101
SEARCH FOR PERTINENT RULE AT FIREWALL
102
104
103
RULE ACTION = REFER TO ACCESS CONTROL PROXY?
NO
PERFORM DROP OR PASS RULE ACTION
YES
105
SEND PACKET TO ACCESS CONTROL PROXY
ANALYZE CONTENT OF PAYLOAD
106
SELECT PERTINENT ACCESS RULE
107
108
IMPLEMENT ACCESS RULE

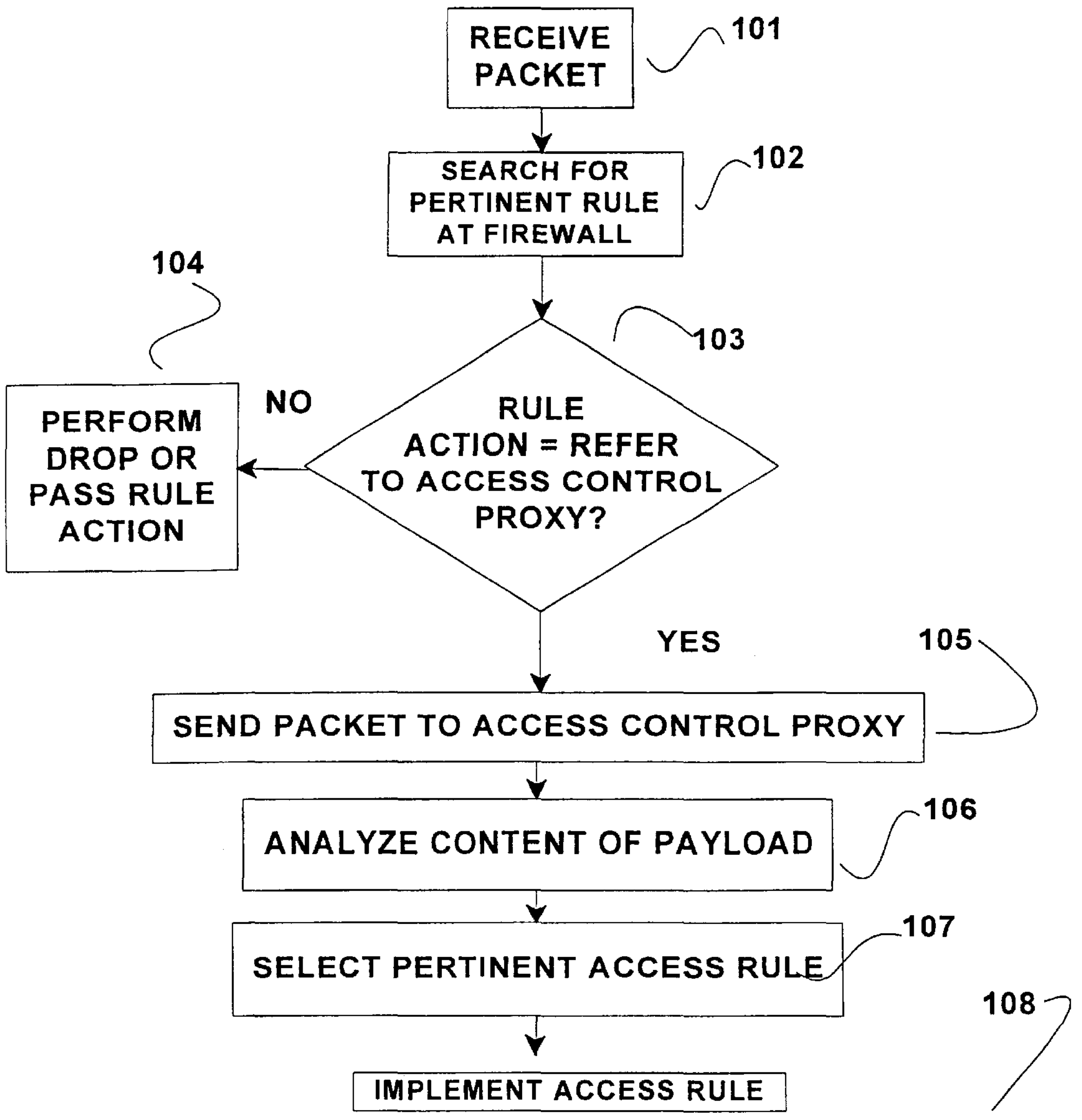

RECEIVE PACKET
101
SEARCH FOR PERTINENT RULE AT FIREWALL
102
104
103
PERFORM DROP OR PASS RULE ACTION
NO
RULE ACTION = REFER TO ACCESS CONTROL PROXY?
YES
105
SEND PACKET TO ACCESS CONTROL PROXY
ANALYZE CONTENT OF PAYLOAD
106
SELECT PERTINENT ACCESS RULE
107
108
IMPLEMENT ACCESS RULE

FIG 1

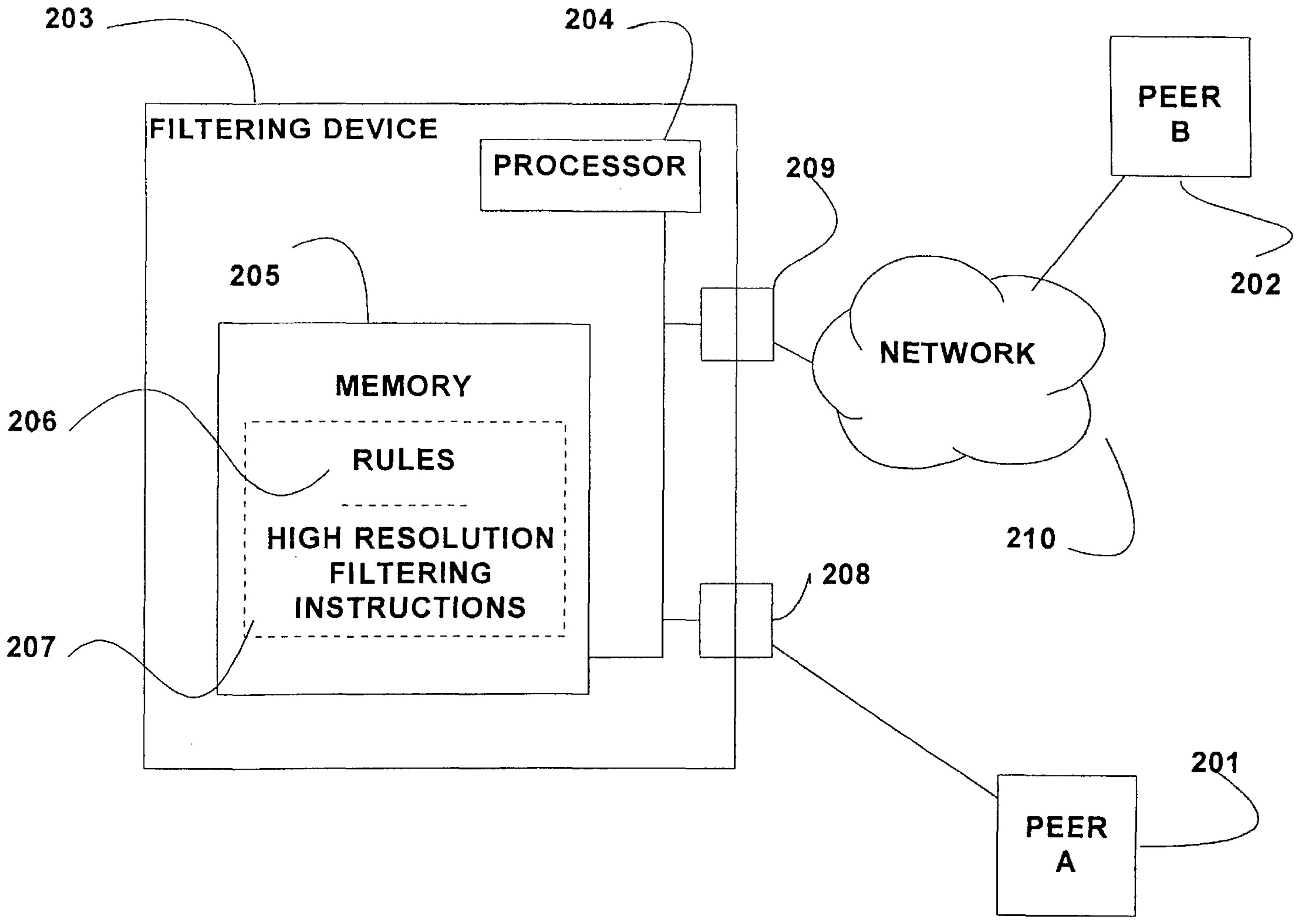

203
204
FILTERING DEVICE
PROCESSOR
209
PEER B
202
205
MEMORY
NETWORK
206
RULES
HIGH RESOLUTION FILTERING INSTRUCTIONS
210
208
207
201
PEER A

FIG 2

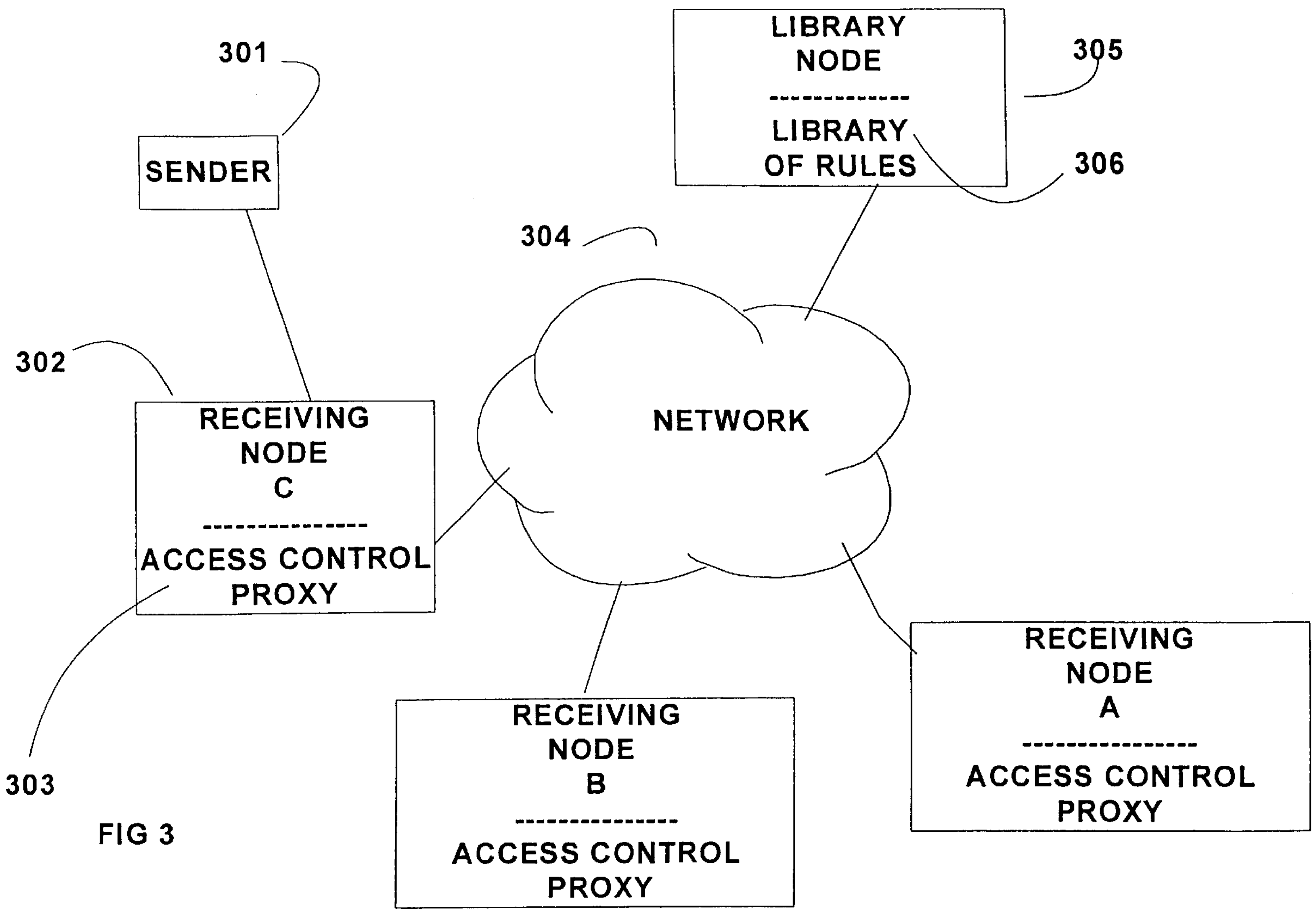

301
SENDER
302
RECEIVING NODE C
ACCESS CONTROL PROXY
303
304
NETWORK
LIBRARY NODE
LIBRARY OF RULES
305
306
RECEIVING NODE B
ACCESS CONTROL PROXY
RECEIVING NODE A
ACCESS CONTROL PROXY

FIG 3

HIGH RESOLUTION ACCESS CONTROL

CROSS REFERENCE TO RELATED APPLICATION

This application claims priority to provisional application 60/105,188 entitled "HIGH RESOLUTION ACCESS CONTROL," filed Oct. 22, 1998, the contents of which are incorporated herein by reference.

FIELD OF THE INVENTION

The field of the invention is information systems access control, and in particular high resolution filtering of packetized information.

BACKGROUND OF THE INVENTION

A firewall regulates the flow of packetized information. A packet includes a header and a payload. The header includes header parameters, including a source and destination address for the packet, as well as source and destination port numbers and a protocol number. Other examples of header parameters include various flags (e.g., security features implemented with respect to the packet (AUTHENTICATED, ENCRYPTED), quality of service requirements (e.g., HIGH, MEDIUM, LOW) for handling the packet, a priority parameter for handling the packet (e.g., ROUTINE, URGENT, FLASH), etc.) The payload includes the data meant to be conveyed by the packet from its source to its intended destination.

A known firewall is placed between the packet's source and intended destination, where it intercepts the packet. The known firewall filters a packet based upon the packet's header parameters and a rule loaded into the firewall. The rule correlates a pattern in the header of a packet with a prescribed action, either PASS or DROP. The filter identifies the rule that applies to the packet based upon the packet's header, and then implements the rule's prescribed action. When a DROP action is performed, the packet is blocked (deleted), and does not reach its intended destination. When a PASS action is performed, the packet is passed on toward its intended destination. The set of rules loaded into a firewall reflect a security policy, which prescribes what type of information is permissible to pass through the firewall, e.g., from which source, to which destination, for which applications, etc.

The set of rules loaded into a known firewall operate at a low level of resolution. As described above, a firewall rule prescribes a PASS or DROP action based only upon the header parameters of the packet. Packet header parameters alone do not reveal the ultimate target of, for example, a connection request from a sender to a destination host. For example, a HyperText Transfer Protocol (HTTP) connection request to send the file located at http://www.att.com/secret.html is not entirely disclosed in the header of the packet initiating the request. The header reveals the Internet Protocol (IP) address of the proxy corresponding to the domain name att.com. However, information regarding the particular file that is being requested, secret.html, is embedded in the payload of the packet. Since known firewalls only filter packets based upon their header parameters, known filters cannot PASS or DROP a packet on the basis of a particular file at a given destination. The same shortfall in known filters exists for filtering a packet destined for a particular newsgroup, chat session, e-mail address, etc.

SUMMARY OF THE INVENTION

The present invention provides high resolution access control for packetized information. In accordance with one embodiment of the present invention, a packet is received at a firewall and referred to an access control proxy. The access control proxy analyzes the contents of the packet, and identifies an access rule based upon the contents. The action prescribed by the access rule is performed with respect to the packet and any related packets. This advantageously provides for filtering a packet based not only upon its header information, as in known firewalls, but upon the information contained in the packet payload.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a flow chart showing the method in accordance with an embodiment of the present invention.

FIG. **2** shows an apparatus in accordance with an embodiment of the present invention.

FIG. **3** shows a system in accordance with the present invention.

DETAILED DESCRIPTION

A flow chart showing the method in accordance with an embodiment of the present invention is shown in FIG. **1**. A packet is received at a firewall, step **101**. The packet has at least one header parameter and a payload. As discussed above, a packet is a discrete unit of information. In one embodiment of the present invention, a packet includes a header and a payload. The header includes header parameters, such as source address, source port, destination address, destination port and protocol number. The payload of the packet includes data being conveyed by the packet, e.g., a connection request, document data, etc. An example of a packet is an Internet Protocol (IP) packet, described in RFC 791, <http://www.library.ucg.ie/CIE/RFC/791/index.htm, visited Sep. 23, 1998>.

After the packet is received, an access rule is identified that corresponds to at least one header parameter of the packet. In one embodiment, this access rule is stored locally at the firewall. In another embodiment, this access rule is obtained from a node external to the firewall.

In accordance with an embodiment of the present invention, the action prescribed by the rule that corresponds to the received packet's header information indicates that the packet is to be referred to an access control proxy. In one embodiment, the access control proxy is specific to a single protocol, e.g., the file transfer protocol (FTP), the hypertext transfer protocol (HTTP), newsgroup protocol, etc.

The access control proxy selects an access rule based upon the contents of the packet. In one embodiment, the access rule is stored locally at the firewall. In another embodiment, the access rule is retrieved from a node external to the firewall. In one embodiment, the access rule is selected based upon the name of the requested file. In another embodiment, it is selected on the basis of the URL of the requested information. For example, an access rule can be selected based upon the domain name of the requested information, or the nth degree domain name of a URL in a packet payload. The "nth degree domain name" is defined as follows: a domain name is comprised of text strings separated by periods, e.g., a.b.c.d.e. The rightmost string (e.g., "e" in the example) is the first degree domain name, the string immediately to the left on the other side of the period is the second degree domain name (e.g., "d" in the example), and each string further to the left is incremented by one degree. Thus, "c" is the third degree domain name, "b" is the fourth degree, etc.

After selecting the access rule based upon the contents of the packet, the access rule is implemented for that packet and any related packets. A related packet, for example, is another packet in the same session request as the first packet. For example, a session is likely to include many packets. The packet or packets that contain sufficient payload information for the access proxy to select a corresponding access rule will be PASSED or DROPPED in accordance with the selected access rule, as will any other packets that comprise the connection request.

This process is shown in more detail in FIG. **1**. A packet is received, step **101**. The set of rules stored at the firewall is searched for a rule that pertains to the header parameters of the packet, step **102**. When such a rule is identified, it is determined if the prescribed action of the rule is to refer the packet to an access control proxy, step **103**. If the prescribed action is not to refer the packet, the action is to PASS or DROP the packet, which is performed for the packet, step **104**. If the prescribed action is to refer the packet, the packet is then sent to the access control proxy, step **105**. In one embodiment, the access control proxy analyzes the content of the packet payload to determine details not available from the header parameters as to the information which the payload requests, step **106**. In another embodiment, the access control proxy analyzes the contents of a plurality of received packets to determine details pertaining to a request for information that is constituted by the plurality of payloads. The number of packet analyzed is sufficient to select an access rule pertaining to the detailed information request, i.e., to decide whether to PASS or DROP the packets pertinent to the request.

The access control proxy then selects an access rule pertaining to the detailed information request contained in the packet payload, step **107**. For example, an access rule prescribes a DROP action for any packet that requests the file located at http://www.att.com/secret.html. On the other hand, an access rule prescribes a PASS action for any packet that requests the file located at http://www.att.com/public.html.

In one embodiment of the present invention, the access control proxy selects an access rule that pertains to the packet based both on an analysis of the payload and the header parameters of the packet. For example, the source address of the packet is included in the header as a header parameter. In one embodiment, the access control proxy selects an access rule that prescribes a DROP action for any packet that requests the file http://www.att.com/secret.html and whose header indicates the packet is from SOURCE A, whereas another selected access rule prescribes a PASS action for any packet that requests the same file, but whose header indicates the packet is from SOURCE B.

In one embodiment of the present invention, the access control proxy then implements the selected access rule for the packet, performing either a PASS or a DROP action with respect to the packet, in accordance with the access rule, step **108**.

An apparatus in accordance with an embodiment of the present invention is shown in FIG. **2**. Peer A **201** (the sender) sends a packet of information addressed to destination Peer B **202** (the destination) through filtering device **203**. The packet payload includes an identifier of a file (e.g., a filename and directory information) requested by peer A **201** and stored at peer B **202**. Filtering device **203** comprises a processor **204**, a memory **205** that stores rules **206** (e.g., both rules that refer a packet to the access control proxy and access rules that are selected by the access control proxy) and high resolution filtering instructions **207** adapted to be executed by processor **204** to perform steps of the method in accordance with an embodiment of the present invention. The filtering device **203** also includes a first port **208** through which the packet is received from Peer A **201**, and a second port **209** through which the packet will pass to Peer B **202** through network **210** if the pertinent rule prescribes a PASS action with respect to the packet.

Peers **201** and **202** are each a computer with a permanent or temporary network address. Network **210** is any information systems network across which the information in the packet can be sent. Examples of network **210** include the Internet, an intranet, a virtual private network, etc.

In one embodiment, processor **204** is a general purpose microprocessor, such as the Pentium II microprocessor manufactured-by the Intel Corporation of Santa Clara, Calif. In another embodiment, processor **204** is an Application Specific Integrated Circuit (ASIC), which has been specifically designed to perform at least some of the steps of the method in accordance with an embodiment of the present invention. ASICs are well-known in the art for application such as digital signal processing. In an embodiment of the present invention that includes an ASIC, at least part of the high resolution filtering instructions **207** can be implemented in the design of the ASIC.

Memory **205** can be Random Access Memory (RAM), a hard disk, a floppy disk, an optical digital storage medium, or any combination thereof. Memory **205** is meant to encompass any means for storing digital information.

High resolution filtering instructions **207** are adapted to be executed by processor **204** to receive a packet, refer the packet to an access control proxy, select an access rule base upon the contents of the payload of the received packet, and then implement the access rule by performing the action (typically PASS or DROP) prescribed by the selected rule with respect to a packet. The term “high resolution filtering instructions” is meant to include access control proxy instructions. In one embodiment, the access rule is retrieved based upon a combination of the contents and header parameters of the packet. In another embodiment, the access rule is selected based upon the contents of one or several packet payloads.

In one embodiment of the present invention, high resolution filtering instructions **207** include firewall instructions and access control proxy instructions. In one embodiment, the firewall instructions are executed on processor **204** as a firewall process, and the access control proxy instructions are executed on processor **204** as an access control proxy process. When filtering device **203** receives a packet, the firewall process searches for and identifies a rule pertinent to the packet. The rule prescribes an action, either PASS, DROP or to REFER the packet to an access control proxy. In one embodiment of the present invention, there is a distinct access control proxy for each different protocol to which a packet can conform, e.g., HTTP, FTP, e-mail, newsgroup, telnet, etc. The protocol of a packet in one embodiment is indicated as a protocol number in the packet header. An embodiment of the present invention advantageously uses the protocol number in the header to refer a packet to the correct access control proxy process.

When a packet is referred to an access control proxy process, the proxy process analyzes the contents of the packet and selects an access rule based upon the results the content analysis. In one embodiment, the selected access rule is stored locally. In another embodiment, the selected access rule is retrieved from an external database. In yet another embodiment, the access rule is dynamically formulated by the proxy. The access rule is implemented at the firewall.

In one embodiment of the present invention, several (more than one) packets are referred to the access control proxy process. The access control proxy process analyzes the contents of the several packets, and selects an access rule based upon the results of this analysis. In one embodiment, the information needed to select an access rule is spread across the contents of the several packets, and may not be contained in any one of the several packets alone. Thus, in one embodiment, the contents of a packet may be represented as:

Packet: SELECT_RULE_**1432**

This shows that there is sufficient information in the single packet to identify the rule that should be selected. On the other hand, consider four packets that contain the following information:

Packet 1: SELECT_RULE_FIRST_DIGIT_**1**
Packet 2: SELECT_RULE_SECOND_DIGIT_**4**
Packet 3: SELECT_RULE_THIRD_DIGIT_**3**
Packet 4: SELECT_RULE_FOURTH_DIGIT_**2**

The above example is primarily heuristic. Another example arises when several packets need to be analyzed to determine what type of message is being carried by the packets, and where traffic is regulated through the firewall based upon the type of message being carried.

In one embodiment, there are a plurality of ports to and from numerous destinations. The port or ports that communicate packets to and from filtering device **203** are meant to encompass any number or configuration of ports. The port configuration is expected to vary to suit the particular connectivity required of a filtering device **203** in a given situation, i.e., in a given context or architecture in which parties communicate through filtering device **203**.

In various embodiments, the functions of the present invention are performed on separate nodes. In one embodiment shown in FIG. **3**, a packet is received from a sender **301** at one of a plurality of receiving nodes **302**, which node **302** then refers the packet to a locally executing access control proxy **303**. If the local access control proxy **303** does not store a rule corresponding to the contents of the packet, it sends a query through network **304** to another separate node **305** that can advantageously function as a central library that stores a large number of access rules **306**, only some of which may be needed at any one time by the plurality of receiving nodes **302**. The library node **305** identifies the pertinent access rule from its collection of access rules **306**, and then sends it to the access control proxy at the requesting

receiving node **302**, which then implements it. This illustrates the advantageous scalability of the present invention. Only relatively few library sites (in relation to the number of receiving nodes) need store large numbers of access rules.

In another embodiment, the firewall is on a receiving node **302**, and performs firewall functions, including receiving a packet (using a rule), referring the packet to the access control proxy, and implementing an access rule. The access control proxy is on another node **305**, and there performs proxy functions including analyzing the packet and selecting an access rule, which it then sends to the receiving node **302** to implement. In other words, the firewall functions can be performed by a different processor than processor that performs the proxy functions.

A medium that stores instructions adapted to be executed on a processor, like memory **205**, is meant to encompass any medium capable of storing digital information. Examples of a medium that stores instructions include a hard disk, a floppy disk, a Compact Disk Read Only Memory (CD-ROM), magnetic tape, flash memory, etc.

The term "instructions adapted to be executed" is meant to encompass more than machine code. The term "instructions adapted to be executed" is meant to encompass source code, assembler, and any other expression of instructions that may require preprocessing in order to be executed by processor. For example, also included is code that has been compressed or encrypted, and must be uncompressed and/or unencrypted in order to be executed by a processor.

The present invention advantageously provides a more efficient, flexible and scalable system and method for implementing the rules of a security policy or policies at a filtering device, because a rule is only loaded at the filtering device when the rule is needed.

What is claimed is:

1. A method for filtering a packet, including the steps of:

a. receiving a packet having at least one header parameter and a payload;

b. selecting an access rule based upon the contents of the payload of the packet received in step a;

c. implementing the access rule for a packet, wherein the access rule is selected based upon a combination of the contents of the packet received in step a and the contents of at least one other packet.

* * * * *

EXHIBIT C

US006715084B2

(12) **United States Patent**
Aaron et al.

(10) **Patent No.: US 6,715,084 B2**
(45) **Date of Patent: Mar. 30, 2004**

(54) FIREWALL SYSTEM AND METHOD VIA FEEDBACK FROM BROAD-SCOPE MONITORING FOR INTRUSION DETECTION

(75) Inventors: **Jeffrey A. Aaron**, Atlanta, GA (US); **Thomas Anschutz**, Conyers, GA (US)

(73) Assignee: **BellSouth Intellectual Property Corporation**, Wilmington, DE (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 28 days.

(21) Appl. No.: **10/108,078**

(22) Filed: **Mar. 26, 2002**

(65) **Prior Publication Data**

US 2003/0188191 A1 Oct. 2, 2003

(51) **Int. Cl.**^{7} **G06F 11/30**; G06F 12/14; H04L 9/00
(52) **U.S. Cl.** **713/201**; 713/200; 709/235
(58) **Field of Search** 713/200, 201

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,621,889 A		4/1997	Lermuzeaux et al.	395/186
5,784,569 A	*	7/1998	Miller et al.	709/235
5,826,014 A		10/1998	Coley et al.	395/187.01
5,892,903 A		4/1999	Klaus	395/187.01
5,931,946 A		8/1999	Terada et al.	713/201
5,991,881 A		11/1999	Conklin	713/201
6,026,502 A		2/2000	Wakayama	714/38
6,061,798 A		5/2000	Coley et al.	713/201
6,119,236 A	*	9/2000	Shipley	713/201
6,134,664 A		10/2000	Walker	713/201
6,167,358 A		12/2000	Othmer et al.	702/188
6,205,551 B1	*	3/2001	Grosse	713/201
6,321,338 B1	*	11/2001	Porras et al.	713/201
6,405,318 B1	*	6/2002	Rowland	713/200
6,460,141 B1	*	10/2002	Olden	713/201
6,513,122 B1	*	1/2003	Magdych et al.	713/201

OTHER PUBLICATIONS

Julia Allen et al, "State of the Practice of Instrusion Detection Technologies" Jan. 2000, Carnegie Mellon University, pp. 1–220.*
Julia Allen et al, "A Safe Bet Cert Cercurity Practices" Summer 2001, IAnewsletter, vol. 4, No. 3, pp. 5–7.*
Security Focus HOME Tools Archive, wysiwg://22/http://www.security–portal.com/tools/categor.*

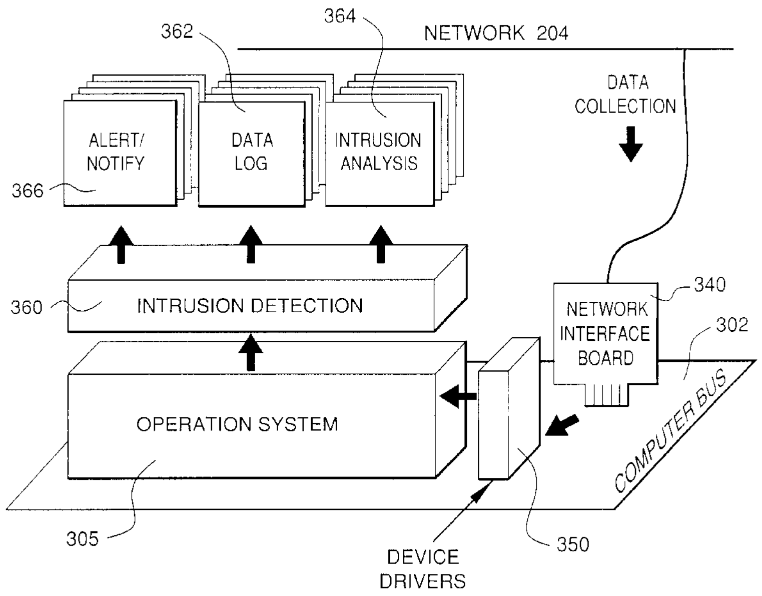
* cited by examiner

Primary Examiner—Ly V. Hua
(74) *Attorney, Agent, or Firm*—Woodcock Washburn LLP

(57) **ABSTRACT**

A broad-scope intrusion detection system analyzes traffic coming into multiple hosts or other customers' computers or sites. This provides additional data for analysis as compared to systems that just analyze the traffic coming into one customer's site. Additional detection schemes can be used to recognize patterns that would otherwise be difficult or impossible to recognize with just a single customer detector. Standard signature detection methods can be used. Additionally, new signatures can be used based on broad-scope analysis goals. An anomaly is detected in the computer system, and then it is determined which devices or devices are anticipated to be affected by the anomaly in the future. These anticipated devices are then alerted to the potential for the future anomaly. The anomaly can be an intrusion or an intrusion attempt or reconnaissance activity.

33 Claims, 5 Drawing Sheets



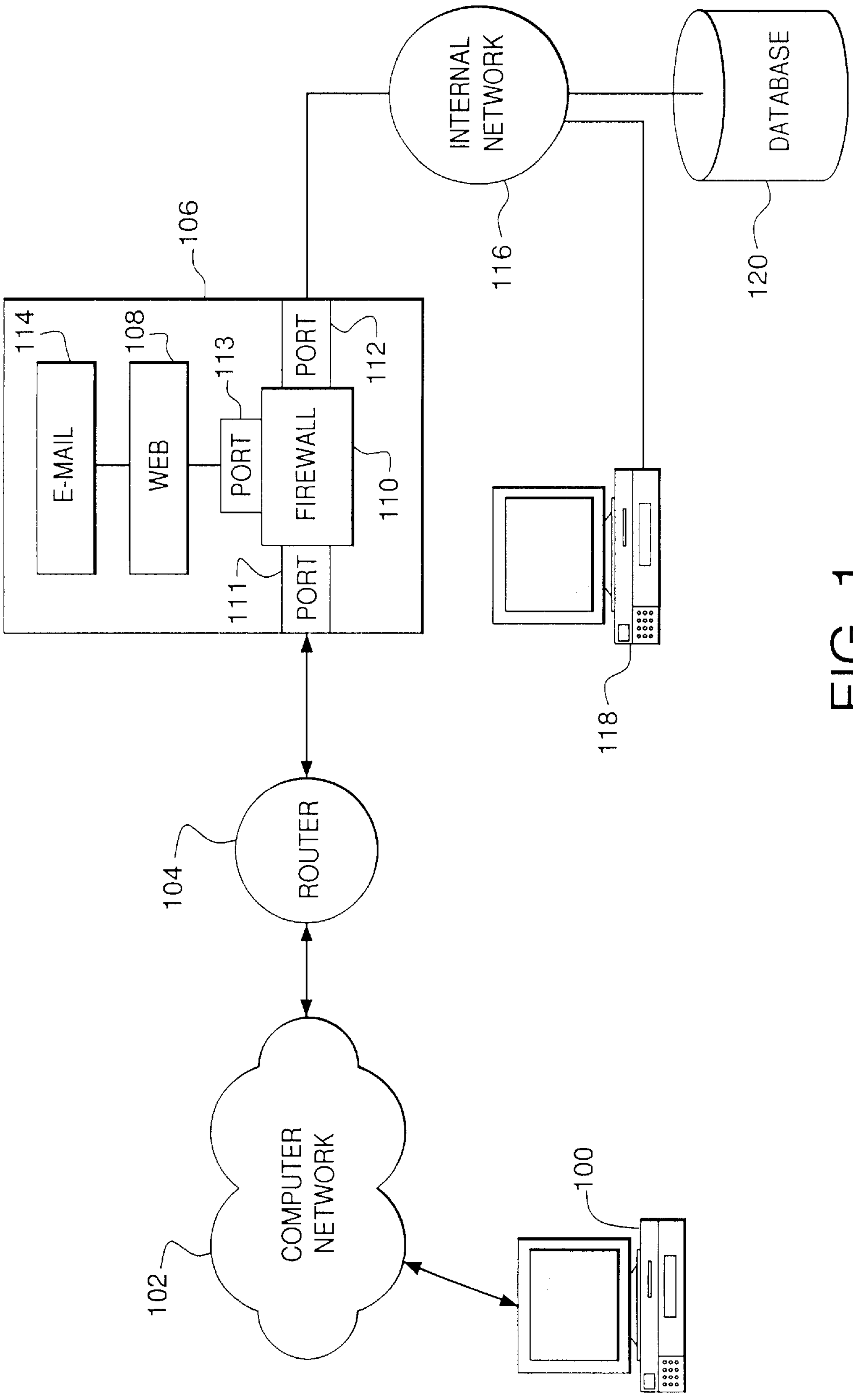


FIG. 1
(PRIOR ART)

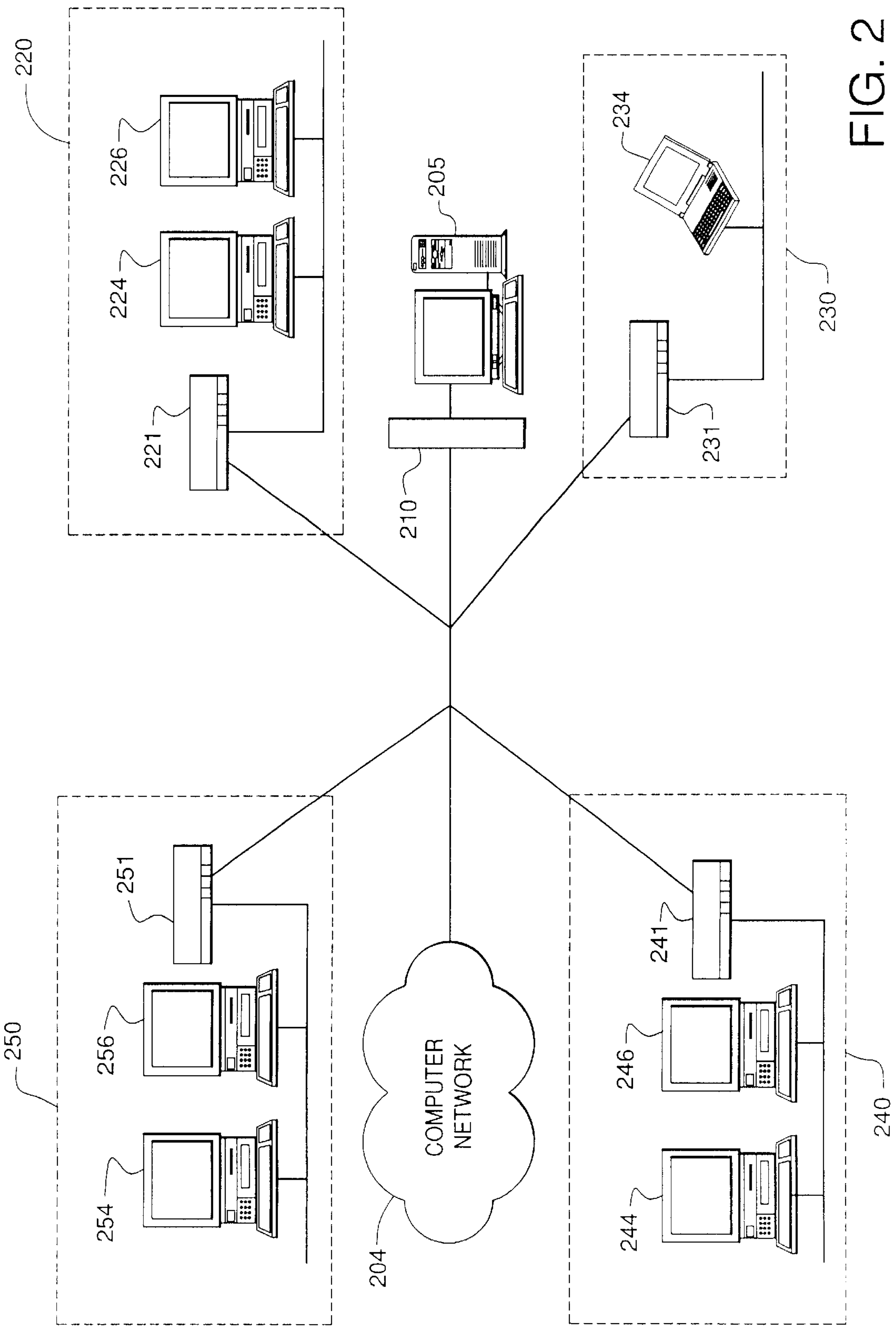

220
226
224
221
205
210
234
230
231
COMPUTER NETWORK
204
250
256
254
251
240
246
244
241

FIG. 2

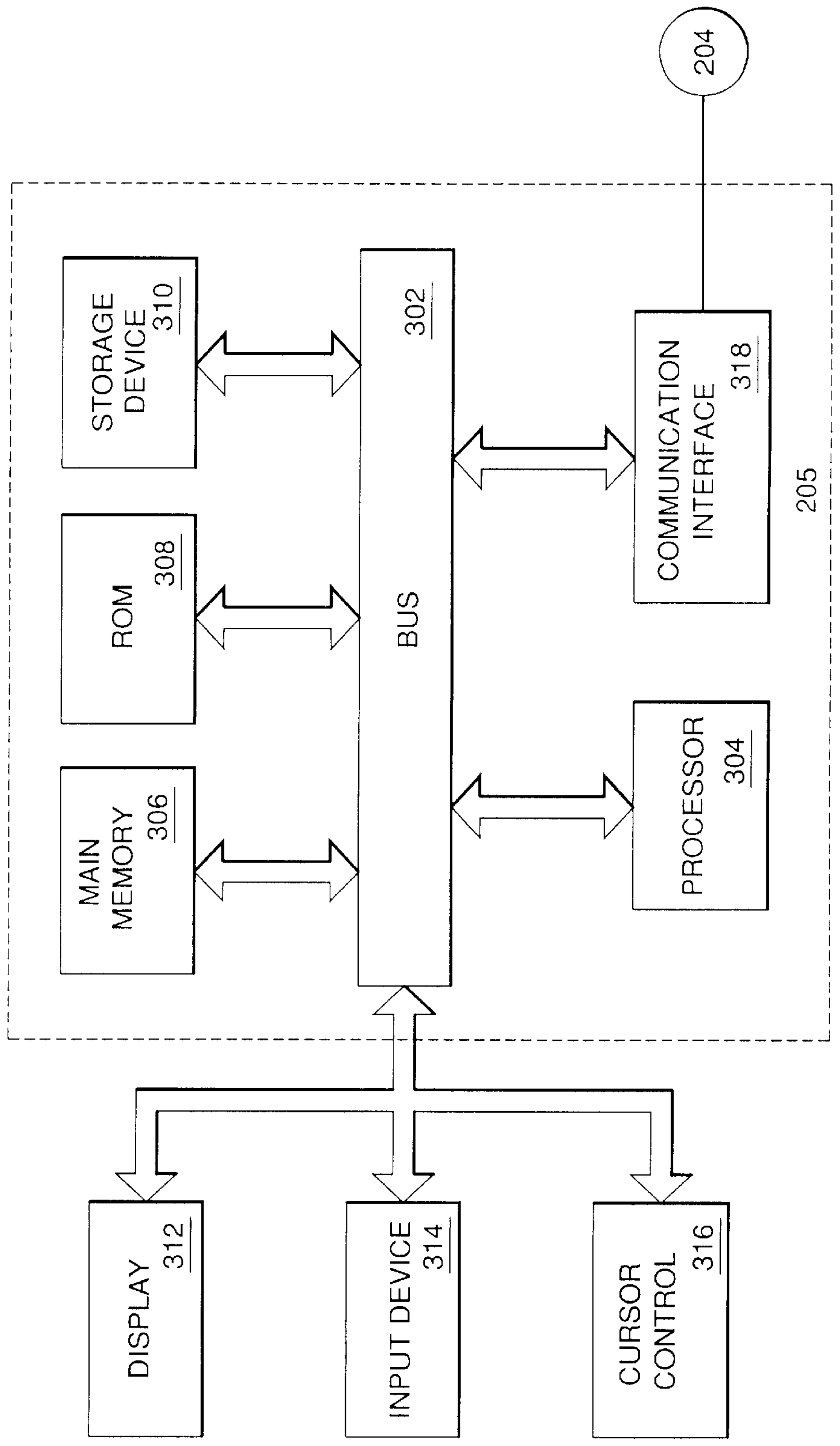

204
STORAGE DEVICE 310
ROM 308
MAIN MEMORY 306
BUS 302
COMMUNICATION INTERFACE 318
PROCESSOR 304
205
DISPLAY 312
INPUT DEVICE 314
CURSOR CONTROL 316

FIG. 3

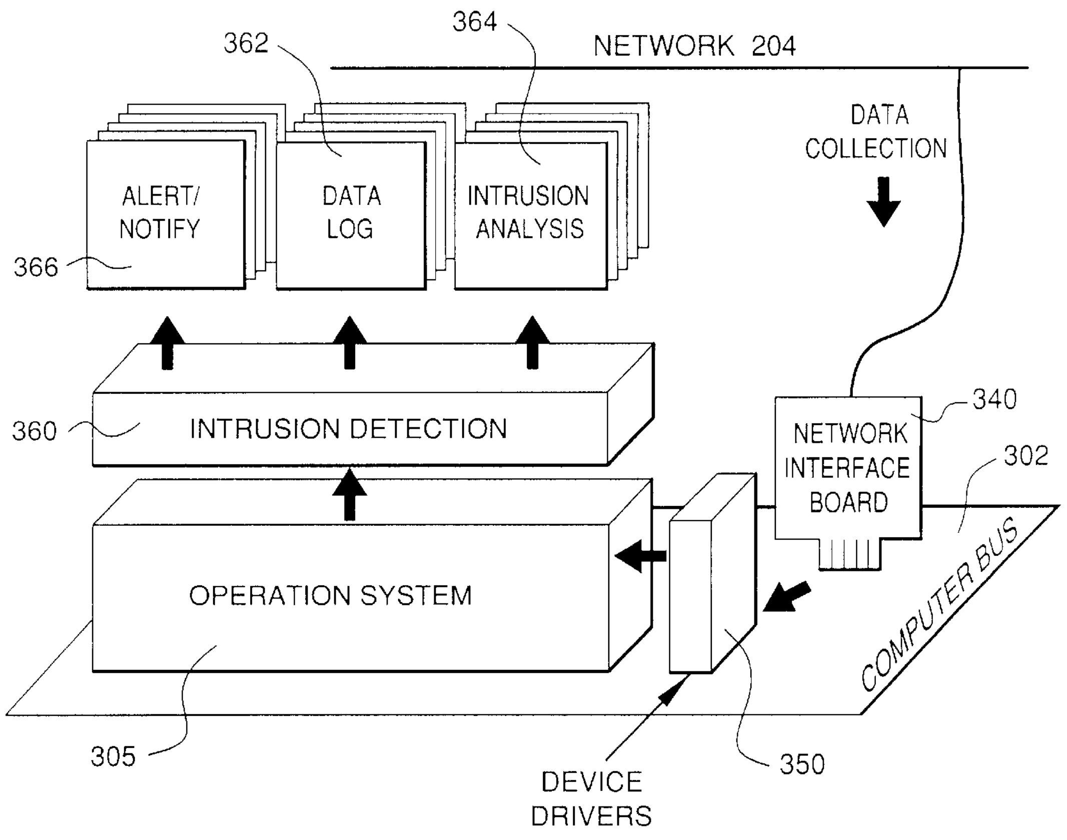

364
362
NETWORK 204
DATA COLLECTION
ALERT/ NOTIFY
DATA LOG
INTRUSION ANALYSIS
366
360
INTRUSION DETECTION
340
NETWORK INTERFACE BOARD
302
OPERATION SYSTEM
COMPUTER BUS
305
DEVICE DRIVERS
350

FIG. 4

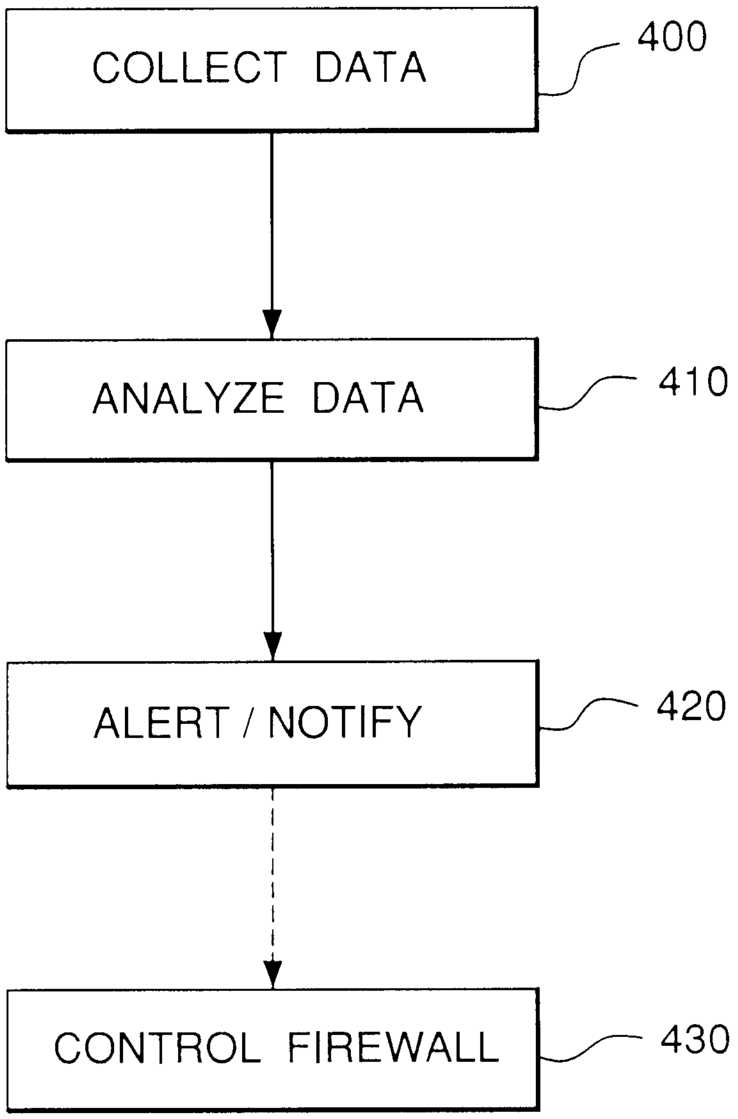

COLLECT DATA
400
ANALYZE DATA
410
ALERT / NOTIFY
420
CONTROL FIREWALL
430

FIG. 5

FIREWALL SYSTEM AND METHOD VIA FEEDBACK FROM BROAD-SCOPE MONITORING FOR INTRUSION DETECTION

FIELD OF THE INVENTION

The present invention relates in general to intrusion detection systems for computer systems and, more particularly, to network-based intrusion detection systems.

BACKGROUND OF THE INVENTION

Numerous present-day computer installations, be they provided with centralized processor units or be they organized in networks interconnecting geographically distributed processor units, have various access points for serving their users. The number of such points and the ease with which they are often accessible have the drawback of facilitating attempts at intrusion by people who are not authorized users and attempts by users of any kind, whether acting alone or in concert, to perform computer operations which such users should not be capable of performing legitimately. These unauthorized users are typically called “hackers” or “crackers”.

Moreover, the open network architecture of the Internet permits a user on a network to have access to information on many different computers, and it also provides access to messages generated by a user’s computer and to the resources of the user’s computer. Hackers present a significant security risk to any computer coupled to a network where a user for one computer may attempt to gain unauthorized access to resources on another computer of the network.

In an effort to control access to a network and, hence, limit unauthorized access to computer resources available on that network, a number of computer communication security devices and techniques have been developed. One type of device which is used to control the transfer of data is typically called a “firewall”. Firewalls are routers which use a set of rules to determine whether a data message should be permitted to pass into or out of a network before determining an efficient route for the message if the rules permit further transmission of the message.

One fundamental technique used by firewalls to protect network elements is known as “packet filtering”. A packet filter may investigate address information contained in a data packet to determine whether the source machine, from which the packet originated, is on a list of allowed addresses. If the address is on the list, the packet is allowed to pass. Otherwise the packet is dropped. Packet filtering using lists of allowed protocols (e.g., file transfer FTP, web access HTTP, email POP) is also sometimes done, either alone or in combination with the more stringent address-based packet filtering method.

One problem with address-based packet filtering is that hackers have developed a technique known as “address spoofing” or “P spoofing” wherein address information within a fabricated packet is manipulated to bypass a packet filter (e.g., by placing the address information of a machine which is on the allowed list within the packet, even though the true source address which would normally be placed within the packet is different and disallowed). Address spoofing may also be used to make it appear that the packet originates in the network that the firewall protects, and thus is on a default allowed list.

An example of a conventional firewall arrangement is depicted in FIG. 1. A host computer **100** communicates with

an institutional computer system **106** over a public network **102** through a router **104**. A router is a network element that directs a packet in accordance with address information contained in the packet. The institutional computer system **106** supports a variety of applications including a Web server **108**, and an e-mail system **114**. A firewall system **110** with ports **111**, **112**, **113** is placed between the router **104** and the institutional computer **106**. Port **112** connects an internal network **116** to the firewall **110**, while ports **111** and **113** connect the public network **102** and the institutional computer **106**, respectively. The internal network **116** may support communication between internal terminal(s) **118** and a database **120**, possibly containing sensitive information. Such a firewall system **110**, however, although intended to protect resources **118** and **120** connected to the internal network **116**, is subject to attack in many ways.

A hacker operating the host computer **100** can utilize publicly accessible applications on the institutional computer system **106**, such as the Web server **108** or the e-mail system **114**, to attack the firewall system **110** or connect to the internal network port **112**. The Web server **108** or the e-mail system **114** may have authority to attach to and communicate through the firewall system **110**. The hacker might be able to exploit this by routing packets through, or mimicking these network elements, in order to attach to, attack, or completely bypass, the firewall system **110**.

Most conventional firewalls, unless configured otherwise, are transparent to packets originating from behind the firewall. Hence, the hacker may insert a source address of a valid network element residing behind the firewall **110**, such as the terminal **118**, to a fictitious packet. Such a packet may then be able to pass through the firewall system **110**. The hacker may even set the packet to be configured to contain a message requesting the establishment of a session with the terminal **118**. The terminal **118** typically performs no checking itself, instead relying on the firewall, and assumes that such a session request is legitimate. The terminal **118** acknowledges the request and sends a confirmation message back through the firewall system **110**. The ensuing session may appear to be valid to the firewall system **110**.

The hacker can also initiate multiple attempts to attach to the port **111**. Technically, a connection to the port is formed before the firewall **110** is able to filter the authority of the request. If enough connection requests hit the port **112**, it may be rendered unavailable for a period of time, denying service to both incoming requests from the public network, and more importantly, denying access to the internal network **116** for outgoing messages. It is readily apparent that conventional firewall systems, such as the one depicted in FIG. **1**, are unacceptably vulnerable in many ways.

Hackers have also developed other ways which may be helpful in bypassing the screening function of a router. For example, one computer, such as a server on the network, may be permitted to receive sync messages from a computer outside the network. In an effort to get a message to another computer on a network, a hacker may attempt to use source routing to send a message from the server to another computer on the network. Source routing is a technique by which a source computer may specify an intermediate computer on the path for a message to be transmitted to a destination computer. In this way, the hacker may be able to establish a communication connection with a server through a router and thereafter send a message to another computer on the network by specifying the server as an intermediate computer for the message to the other computer.

In an effort to prevent source routing techniques from being used by hackers, some routers (including some firewalls) may be configured to intercept and discard all source routed messages to a network. For a router configured with source routing blocking, the router may have a set of rules for inbound messages, a set of rules for outbound messages and a set of rules for source routing messages. When a message which originated from outside the network is received by such a router, the router determines if it is a source routed message. If it is, the router blocks the message if the source routing blocking rule is activated. If blocking is not activated, the router allows the source routed message through to the network. If the message is not a source routed message, the router evaluates the parameters of the message in view of the rules for receiving messages from sources external to the network. However, a router vulnerability exists where the rules used by the router are only compared to messages that are not source routed and the source routed blocking rule is not activated. In this situation, the router permits source routed messages through without comparing them to the filtering rules. In such a case, a computer external to the network may be able to bypass the external sync message filter and establish a communication connection with a computer on the network by using source routed messages.

A typical secure computer network has an interface for receiving and transmitting data between the secure network and computers outside the secure network. A plurality of network devices are typically behind the firewall. The interface may be a modem or an Internet Protocol (IP) router. Data received by the modem is sent to a firewall. Although the typical firewall is adequate to prevent outsiders from accessing a secure network, hackers and others can often breach a firewall. This can occur by a variety of methods of cyber attack which cause the firewall to permit access to an unauthorized user. An entry by an unauthorized computer into the secured network, past the firewall, from outside the secure network is called an intrusion. This is one type of unauthorized operation on the secure computer network.

There are systems available for determining that a breach of computer security has occurred, is underway, or is beginning. These systems can broadly be termed "intrusion detection systems". Existing intrusion detection systems can detect intrusions and misuses. The existing security systems determine when computer misuse or intrusion occurs. Computer misuse detection is the process of detecting and reporting uses of processing systems and networks that would be deemed inappropriate or unauthorized if known to responsible parties, administrators, or owners. An intrusion is an entry to a processing system or network by an unauthorized outsider.

Misuse detection and reporting research has followed two basic approaches: anomaly detection systems and expert systems.

Anomaly detection systems look for statistically anomalous behavior. Statistical scenarios can be implemented for user, dataset, and program usage to detect "exceptional" use of the system. Since anomaly detection techniques do not directly detect misuse, they do not always detect most actual misuses. The assumption that computer misuses would appear statistically anomalous has been proven unreliable. When recordings or scripts of known attacks and misuses are replayed on computers with statistical anomaly detection systems, few if any of these scripts are identified as anomalous. This occurs for a variety of reasons which reduce the indirect detection accuracy.

In general, anomaly detection techniques cannot detect particular instances of misuses unless the specific behaviors

associated with those misuses also satisfy statistical tests (e.g., regarding network data traffic or computer system activity) without security relevance. Anomaly detection techniques also produce false alarms. Most of the reported anomalies are purely coincidental statistical exceptions and do not reflect actual security problems. These false alarms often cause system managers to resist using anomaly detection methods because they increase the processing system workload and need for expert oversight without substantial benefits.

Another limitation with anomaly detection approaches is that user activities are often too varied for a single scenario, resulting in many inferred security events and associated false alarms. Statistical measures also are not sensitive to the order in which events occur, and this may prevent detection of serious security violations that exist when events occur in a particular order. Scenarios that anomaly detection techniques use also may be vulnerable to conscious manipulation by users. Consequently, a knowledgeable perpetrator may train the adaptive threshold of detection system scenarios over time to accept aberrant behaviors as normal. Furthermore, statistical techniques that anomaly detection systems use require complicated mathematical calculations and, therefore, are usually computationally expensive.

Expert systems (also known as rule-based systems) have had some use in misuse detection, generally as a layer on top of anomaly detection systems for interpreting reports of anomalous behavior. Since the underlying model is anomaly detection, they have the same drawbacks of anomaly detection techniques. Expert systems attempt to detect intrusions by taking surveillance data supplied by a security system of the computer installation and by applying knowledge thereto relating to potential scenarios for attacking the computer installation. This is not fully satisfactory either, since that method only detects intrusions that correspond to attack scenarios that have previously been stored.

In contrast to the two research approaches, most recent practical attempts at detecting misuse have relied on a signature or pattern-detection mechanism with a signature being the set of events and transitions/functions that define the sequence of actions that form an attack or misuse. A signature mechanism uses network sensors to detect data traffic or audit trail records typically generated by computer operating systems. The designer of the product which incorporates the mechanism selects a plurality of events that together form the signature or the attack or misuse. Although the signature mechanism goes a step beyond expert systems, it is similar to an expert system because it relies upon signatures or rules.

Importantly, intrusion detection methods used today are plagued by false positive events, and the inability to detect the earliest stages of network attacks. Conventional intrusion detection techniques are based on specialized equipment located at a specific customer's premises and hence cannot see the hacker's activities over a broader scale. A need exists for an intrusion detection system which can provide early warning of potential misuses and intrusions with greater knowledge than can be obtained from detection at a single customer's premises. Early warning can be provided by specially examining detection events over a broader scale or scope, i.e., that of many aggregated customers or of the intervening network.

Intrusion detection products and services presently available are directed to the analysis of a single customer's data to determine intrusion events, but lack the capability to perform broad-scope intrusion analysis/detection.

It is readily apparent that the design, implementation, and limitations of conventional firewalls has rendered them highly vulnerable to hacker attack. What is needed is an improved firewall functionality or system that overcomes the foregoing disadvantages and is resistant to hacker attack.

It is also readily apparent that the design, implementation, and limitations of conventional intrusion/misuse detection systems has rendered them unreliable and inefficient. Furthermore, these intrusion detection systems are vulnerable to hacker techniques which render them insensitive to misuse. What is needed is an improved intrusion detection functionality or system that overcomes the foregoing disadvantages and is resistant to hacker attack.

In security, there is a trade-off between safety and other conflicting goals such as usability, usefulness, allowed features, freedom of action, etc. Firewalls currently must be configured non-optimally, i.e., at one extreme of the security trade-off since they cannot react to the current and/or future security environment, and lacking this ability, security must err on the side of safety. Without knowledge of the current (and potentially the expected/predicted) security forecast, the firewall must be configured for the worst-case scenario. But in reality, the security forecast is seldom so extreme. Thus, the firewall should ideally be configured much of the time on a less strict basis, allowing many additional services to be opened through the firewall which, although adding potential vulnerabilities, also add considerable value for the user and the organization/enterprise. However, if this somewhat lax configuration is maintained even in the face of attacks, when the potential vulnerabilities introduced by the presence of the valuable services are much more likely to be exploited, then overall security is lost. So it is desirable for security in this case to have the ability to rapidly respond in the appropriate manner to deteriorating forecast conditions by closing the firewalls (i.e., adding the required firewall filtering) when the situation deteriorates. Feedback to security devices from broad-scope monitoring is needed to make such optimal configuration control/adjustment possible, thereby solving the current problems and thus improving the value of security by avoiding the need for excessive "worst-casebased" restrictions.

SUMMARY OF THE INVENTION

The present invention is directed to a system and method for broad-scope intrusion detection. The system analyzes traffic coming into multiple hosts or other customers' computers or sites. This provides additional data for analysis as compared to systems that just analyze the traffic coming into one customer's site (as a conventional intrusion detection system does). Therefore, additional detection schemes can be used to recognize patterns that would otherwise be difficult or impossible to recognize with just a single customer detector. Standard signature detection methods can be used. Additionally, new signatures and methods/algorithms can be used based on broad-scope analysis goals.

Other embodiments of the present invention are directed to a system and method of alerting a device in a networked computer system comprising a plurality of devices to an anomaly. An anomaly is detected in the computer system, and then it is determined which devices or devices are anticipated to be affected by the anomaly in the future. These anticipated devices are then alerted to the potential for the future anomaly. The anomaly can be an intrusion or an intrusion attempt or reconnaissance activity.

According to aspects of the invention, the devices are polled in a predetermined sequential order, and a device anticipated to be affected by the anomaly is a device that has not been polled.

According to other aspects of the invention, an anomaly warning is transmitted from a first device to a central analysis engine, responsive to detecting the anomaly at the first device. Preferably, the anomaly warning comprises a unique device identifier.

According to further aspects of the invention, detecting the anomaly comprises analyzing a plurality of data packets with respect to predetermined patterns. Analyzing the data packets can comprise analyzing data packets that have been received at at least two of the plurality of devices including the first device.

According to further aspects of the invention, alerting the device comprises alerting a firewall associated with the device that an anomaly has been detected. Moreover, the device that is anticipated to be affected by the anomaly can be controlled (e.g., have its firewall adjusted).

The foregoing and other aspects of the present invention will become apparent from the following detailed description of the invention when considered in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** depicts a computer network arrangement having a conventional firewall arrangement;

FIG. **2** shows in, schematic form, a computer network system including an intrusion detection system in accordance with the present invention;

FIG. **3** is a detailed block diagram of an exemplary computer system with which the present invention can be used;

FIG. **4** shows in block form aspects of the intrusion detection system in accordance with the present invention; and

FIG. **5** shows a flow chart of an exemplary intrusion detection method in accordance with the present invention.

DESCRIPTION OF EXEMPLARY EMBODIMENTS AND BEST MODE

The invention uses components, such as a computer system with a multi-tasking operating system, a network interface card, and network surveillance software, acting together to provide system functionality. This combination of hardware and software attached to a network is described more fully below and will perform the processes described below.

FIG. **2** shows in, schematic form, a computer network-system including an intrusion detection system in accordance with the present invention. A plurality of network devices such as hosts, servers, and personal computers attached within customer site networks (shown here as customer site networks **220**, **230**, **240**, **250**), are shown coupled to an intervening computer network **204**, such as a public network like the Internet. Routers (not shown) are typically used in the coupling. The customer site networks represent "internal" protected networks local to a particular corporation or site, for example. The customer site networks may or may not be publicly accessible or may comprise a publicly accessible network and an internal "private" network. Each customer site network or LAN (Local Area Network) comprises one or more hosts (e.g., customer site network **220** is shown with hosts **224**, **226**; customer site network **230** is shown with host **234**; customer site network **240** is shown with hosts **244**, **246**; and customer site network

250 is shown with hosts **254**, **256**). Each site network is connected to the intervening computer network **204** via a firewall (e.g., host **220** is shown with firewall **221**; host **230** is shown with firewall **231**; host **240** is shown with firewall **241**; and host **250** is shown with firewall **251**).

A firewall connects the network **204** to an internal network. The firewall is a combination hardware and software buffer that is between the internal network and external devices outside the internal computer network. The firewall allows only specific kinds of messages to flow in and out of the internal network. As is known, firewalls are used to protect the internal network from intruders or hackers who might try to break into the internal network. The firewall is coupled to an interface (not shown). The interface is external to the internal network and can be a modem or an Internet Protocol (IP) router and serves to connect the internal network to devices outside the internal network.

A separately maintained data collection and processing center, comprising a computer or server **205** with firewall **210**, is also coupled to the computer network. Although the data collection and processing center is implemented as a network device which is part of a wired local network, it is also envisioned as possibly being connected to the network **204** by a wireless link.

Each network device can be considered a node because each device has an addressable interface on the network. As can be appreciated, many other devices can be coupled to the network including additional personal computers, minimainframes, mainframes and other devices not illustrated or described which are well known in the art.

The system performs broad-scope intrusion detection by monitoring the communications on a network or on a particular segment of the network. The data collection and processing center receives information from the various network devices attached to the computer network **204**. For example, all communications sent to each host **220**, **230**, **240**, **250** are forwarded to, or otherwise captured by, the data collection and processing center. Thus, the data collection and processing center receives all communications (i.e., the data) originating from a user on the computer network **204** and flowing to host **220** (and vice versa), for example, as well as all communications originating from the computer network **204** and flowing to all other hosts (and vice versa).

It should be noted that certain devices can be used as sensors to sense data traffic and pass their findings on to the data collection and processing center or other central processing system, and other separate devices may include computer hosts, firewalls, and other systems which may be the potential targets of attack by a hacker, and/or may be adjusted in response to detected attacks, either manually or automatically.

The present invention is usable on such networks as ARCnet, Ethernets and Token-Ring networks, wireless networks, among other network types. The network, in this example, has a network cable, also known as media, which may be of any known physical configuration including unshielded twisted pair (UTP) wire, coaxial cable, shielded twisted pair wire, fiber optic cable, and the like. Alternatively, the network devices could communicate across wireless links.

The system of the present invention is designed and intended to operate compatibly on networks which communicate using the Transmission Control Protocol/Internet Protocol (TCP/IP) standard, although other communications standards (or even proprietary protocols) could be used. Network TCP/IP data is packetized, and sent in frames which are structured to be compatible with any network device which complies with the TCP/IP standards. A typical frame or packet transmitted across the Internet contains a preamble, destination address, source address, type field, data field, and a cyclical redundancy check (CRC). The preamble contains data used by the communicating computer systems to synchronize or handshake. Destination and source Internet Protocol (IP) addresses represent the principals communicating and the packet type indicates the type of communication. The data field contains the actual information content of the dialogue. The CRC is an integrity check facilitated between the two systems participating in the conversation.

The present invention provides aggregate traffic/intrusion monitoring in the provider network. This allows for a broader scope of network activity to be considered and analyzed, not just relevant to a single customer, but across some or all customers. The additional data is valuable because the probing/reconnaissance activities of would-be intruders typically cover a large number of customers, so as to select those with security weaknesses for more in-depth attack. Additional patterns of broadly suspicious activity can thus be correlated/recognized across many customers.

The present invention uses a multi-stage technique in order to improve intrusion detection efficacy and obtain broader scope detection. First, suspicious network traffic events are collected (potentially in context) and forwarded to a central database and analysis engine, then the centralized engine uses pattern correlations across multiple customer's events in order to better determine the occurrence and sources of suspected intrusion-oriented activity prior to actually alarming. Second, upon detection of suspected reconnaissance and probing, the detection process can adjust its matching parameters and alarm thresholds to focus sensitivity on attacks from suspected sources (hackers) against specific targets (customers). Third, actual occurrence of anticipated attacks against specific targets can be used to adjust the broad-scope matching parameters, providing both positive and negative feedback which selectively adjusts specific pattern sensitivity. This process is different from conventional approaches, in that a broader scope of data is utilized in new ways. It should be noted that, in addition to multistage techniques, the present invention can implement monolithic techniques in which a broad scope of customers' events are correlated at a central analysis engine.

The system analyzes traffic coming into multiple hosts or other customer's computers or sites. This provides additional data for analysis as compared to systems that just analyze the traffic coming into one customer's site (as a typical firewall does). Therefore, additional detection schemes can be used to recognize patterns that would otherwise be difficult or impossible to recognize with just a single customer detector. Standard scanning patterns can be used for the data as well, such as sequential or pseudorandom techniques.

The data collection and processing center collects data from multiple or all the customers and analyzes the data. In this manner, the number of false alarms is decreased (because multiple occurrences of an activity may trigger an alarm, but the present invention can scan a large number of customers, so certain types of harmless activity that otherwise would be perceived as a threat can be viewed and discounted as not a threat). Moreover, predictions can be made about future events that may affect customers in the sequence. Thus, the present invention can be used to block future hacks and determine the source address of the hacker.

The present invention monitors the traffic from a plurality of customers. Different types of algorithms can be used to look for different types of patterns that would not be recognizable by a conventional intrusion detection system at a single customer site. The algorithms preferably reside in a back end data center. Data from existing customer's conventional intrusion detection system is provided to the central database and then analyzed. Data records comprise, for example, a time-stamp, a description of the activity, and the source of the probe.

FIG. **3** is a detailed block diagram of an exemplary computer system **205** of a data collection and processing center with which the present invention can be used. The system includes a bus **302** or other communication mechanism for communicating information, and a processor **304** coupled with the bus **302** for processing information. The system also includes a main memory **306**, such as a random access memory (RAM) or other dynamic storage device, coupled to the bus **302** for storing information and instructions to be executed by processor **304**. Main memory **306** also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor **304**. The system further includes a read only memory (ROM) **308** or other static storage device coupled to the bus **302** for storing static information and instructions for the processor **304**. A storage device **310**, such as a magnetic disk or optical disk, is provided and coupled to the bus **302** for storing information and instructions.

The system **205** may be coupled via the bus **302** to a display **312**, such as a cathode ray tube (CRT) or a flat panel display, for displaying information to a computer user. An input device **314**, including alphanumeric and other keys, is coupled to the bus **302** for communicating information and command selections to the processor **304**. Another type of user input device is cursor control **316**, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor **304** and for controlling cursor movement on the display **312**.

The system **205** also includes a communication interface **318** coupled to the bus **302**. Communication interface **318** provides a two-way data communication as is known. For example, communication interface **318** may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface **318** may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Furthermore, the communication interface **318** may be coupled to the network cable **302**. Wireless links may also be implemented. In any such implementation, communication interface **318** sends and receives electrical, electromagnetic or optical signals which carry digital data streams representing various types of information. Of particular note, the communications through interface **318** permits the transmission or receipt of broad-scope intrusion detection information. The system **205** receives data from each of the nodes being monitored on the network.

The system **205** collects the data, filters the data, and processes the data to provide security indications and warnings.

The processor **304** can execute sequences of instructions contained in the main memory **306**. Such instructions may be read into main memory **306** from another computer-readable medium, such as storage device **310**. However, the computer-readable medium is not limited to devices such as storage device **310**. For example, the computer-readable medium may include a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, a RAM, a PROM, an EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave embodied in an electrical, electromagnetic, infrared, or optical signal, or any other medium from which a computer can read. Execution of the sequences of instructions contained in the main memory **306** causes the processor **304** to perform the process steps described below. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

FIG. **4** shows in block form aspects of the system **205** in accordance with the present invention. The intrusion detection portion of the system receives data from the various intrusion detection systems on the network and analyzes this data to detect an attempted intrusion or an intrusion or reconnaissance activity. The data is logged and analyzed. If an intrusion is detected, an alert is logged.

The broad-scope intrusion monitoring system operates through a computer, attached to the network, in the preferred embodiment by an interface card or network interface board **340**. In the preferred embodiment, the network interface board **340** contains a preset and unique identifier such as an Internet address or a hardware address. The unique address provides the means for an attached computer system to identify intended packets and ignore the rest, as is well known in the art. The system utilizes standard device drivers **350** to forward all packets into the host **205** from the network **204** regardless of the address in the packets. Preferably, the system is transparent and inaccessible to an intruder, thereby preserving the authenticity of the logged entries made by the system. To this end, encryption and authentication means can be used, as known to those skilled in the art.

The system preferably monitors the network traffic substantially in its entirety. Upon receipt of the network packets, the interface board **340** passes the packet and all data contained within to the operating system **305** of the system computer. Once there, it is stored in memory (e.g., memory **306**) awaiting entry to the next phase which is the intrusion detection process **360**, described below. In the intrusion detection process, the data is first logged into a data log **362**. The data is then analyzed **364**, and alerts or notifications **366** are thereafter generated.

The computer equipment configuration which may be used in the preferred embodiment may be, for example, conventional computer running a conventional operating system, available as commercial-off-the-shelf products as known to one skilled in the art.

FIG. **5** shows a flow chart of an exemplary intrusion detection method in accordance with the present invention. At step **400**, data is collected or otherwise received at the data collection and processing center from the sensors coupled to the network, whether they be computers or special-purpose devices. Preferably, the data is collected in a predetermined order from the hosts. At step **410**, the data is analyzed to determine if any intrusions have been (or are being) attempted. At step **420**, if any intrusions or attempted intrusions or reconnaissance activity have been detected, the appropriate alerts or notifications are transmitted to the pertinent administrators of the hosts on the network. In this manner, the administrators, and thereby the hosts for which they are responsible, can be prepared for an incoming intrusion, or can take other precautions against future intrusions, or can check their systems to determine if any access was gained in previous intrusion attempts. Because the data is determined in a predetermined order from the sensors, an intrusion attempt that is detected at an earlier, already polled sensor, can be determined and administrators of other hosts, that have not yet been hit by the intrusion attempt, can be alerted about the possibility of such an intrusion attempt. Thus, the present invention gathers and exploits intrusion monitoring data related to many customers rather than just a single customer, thereby reducing inaccurate declarations of intrusion events and more readily detecting the earliest stages of attempted attacks.

It is contemplated that feedback from the broad-scope intrusion detection system is provided to firewalls, secondary (narrow-scope) intrusion detection system devices, hosts (computers), routers, etc. so that the associated firewalls can adjust in response to expected attacks determined to be forthcoming by the intrusion detection system. Such feedback to customer site devices (of all sorts, and especially the firewalls) is useful to enhance security. Such feedback can also be provided to a service provider's network to further deter the attack.

To prevent this approach from itself being attacked, exploited, or fooled by hackers, secure feedback connectivity could be accomplished using encrypted communication via either specially-designed encrypting methods or tunneled via standards such as IPsec (IETF "IP security" standard) or SSL ("secure sockets layer") or SSH ("secure shell"), which provide authentication and encryption functions to secure the transmitted feedback or "configuration change" data. Via the encrypting protocol or inside the encrypted "tunnel," standard data transfer protocols such as FTP could be used to actually transfer information and SNMP to collect/poll status (additionally or alternately, CORBA objects or JAVA programs or applets could be transferred back and forth). These are exemplary methods, and proprietary protocols rather than standards could also be used. These could be done on virtually any sort of network.

Each device and each type of device being controlled/adjusted/reconfigured preferably has that capability in software, which could be done via a device driver or API (application programming interface) or other technical means which allows control or adjustment. It is contemplated that, in addition to notifying the firewall or other host device of an impending attack, the system could control the firewall or other host device to reconfigure or adjust pertinent parameters in anticipation of the attack, at optional step **430**. For each type of device, the parameters or items controlled/adjusted would be different (e.g., filtering parameters/rules for firewalls, allowed services and open ports for hosts, detection parameters or "extent of detection" parameters for intrusion detection system devices, etc.). The present invention provides the ability to detect pre-attack events—this provides lead time to adjust the firewall (or other device) parameters on each of a plurality of hosts before the actual attack occurs. Adjustments after the fact are a less desirable way to maintain security. The broad-scope intrusion detection system algorithms and operation can be adjusted and tuned to specifically gather the information needed to specify the configuration changes/adjustments needed.

Conventional intrusion detection systems merely provide indications of already occurred hacker events and attacks. There is no functionality or capability present in conventional intrusion detection systems to determine near-real-time parameter adjustments for firewalls, etc. which solve the problem. Even if a conventional intrusion detection system was improved so that it could adjust firewall parameters based on what it detects, this adjustment would necessarily happen after the attack, and thus be of little value.

It should be understood that the inventive principles described in this application are not limited to the components or configurations described in this application. It should be understood that the principles, concepts, systems, and methods shown in this application may be practiced with software programs written in various ways, or different equipment than is described in this application without departing from the principles of the invention.

Although illustrated and described herein with reference to certain specific embodiments, the present invention is nevertheless not intended to be limited to the details shown. Rather, various modifications may be made in the details within the scope and range of equivalents of the claims and without departing from the invention.

What is claimed is:

1. A method of alerting at least one device in a networked computer system comprising a plurality of devices to an anomaly, at least one of the plurality of devices having a firewall, comprising:

detecting an anomaly in the networked computer system using network-based intrusion detection techniques comprising analyzing data entering into a plurality of hosts, servers, and computer sites in the networked computer system;

determining which of the plurality of devices are anticipated to be affected by the anomaly by using pattern correlations across the plurality of hosts, servers, and computer sites; and

alerting the devices that are anticipated to be affected by the anomaly.

2. The method of claim **1**, further comprising:

determining which of the plurality of devices have been affected by the anomaly; and

alerting the devices that have been affected by the anomaly.

3. The method of claim **1**, further comprising adjusting the firewall of each of the devices that is anticipated to be affected by the anomaly responsive to the detection of the anomaly.

4. The method of claim **1**, wherein the anomaly comprises one of an intrusion and an intrusion attempt.

5. The method of claim **1**, wherein detecting the anomaly comprises analyzing a plurality of data packets with respect to predetermined patterns.

6. The method of claim **5**, wherein analyzing the data packets comprises analyzing data packets that have been received at at least two of the plurality of devices.

7. The method of claim **1**, wherein detecting the anomaly comprises recognition of an intrusion and further comprising generating an automated response to the intrusion.

8. The method of claim **1**, further comprising adjusting anomaly detection sensitivity and alarm thresholds based on the detected anomaly.

9. A method of alerting a device in a networked computer system comprising a plurality of devices to an anomaly, comprising:

detecting an anomaly at a first device in the computer system using network-based intrusion detection techniques comprising analyzing data entering into a plurality of hosts, servers, and computer sites in the networked computer system;

determining a device that is anticipated to be affected by the anomaly by using pattern correlations across the plurality of hosts, servers, and computer sites; and alerting the device that is anticipated to be affected by the anomaly.

10. The method of claim **9**, wherein the plurality of devices are polled in a predetermined sequential order, the first device being polled prior to detecting the anomaly, and the device anticipated to be affected by the anomaly is a device that has not been polled.

11. The method of claim **9**, further comprising transmitting an anomaly warning from the first device to a central analysis engine, responsive to detecting the anomaly at the first device, the anomaly warning comprising a unique device identifier.

12. The method of claim **9**, wherein the anomaly comprises one of an intrusion and an intrusion attempt.

13. The method of claim **9**, wherein detecting the anomaly comprises analyzing a plurality of data packets with respect to predetermined patterns.

14. The method of claim **13**, wherein analyzing the data packets comprises analyzing data packets that have been received at at least two of the plurality of devices including the first device.

15. The method of claim **9**, wherein alerting the device comprises alerting a firewall associated with the device that the anomaly has been detected.

16. The method of claim **9**, wherein alerting the device comprises generating and transmitting an electronic notification to one of the device and an administrator of the device.

17. The method of claim **9**, further comprising controlling the device that is anticipated to be affected by the anomaly.

18. The method of claim **9**, further comprising adjusting anomaly detection sensitivity and alarm thresholds based on the detected anomaly.

19. An intrusion detection and alerting system for a computer network comprising:

a plurality of devices coupled to the computer network, each device adapted to at least one of: (1) sense data and provide the data to a data collection and processing center, and (2) be adjustable; and

the data collection and processing center comprising a computer with a firewall coupled to the computer network, the data collection and processing center monitoring data communicated to at least a portion of the plurality of devices coupled to the network, detecting an anomaly in the network using network-based intrusion detection techniques comprising analyzing data entering into a plurality of hosts, servers, and computer sites in the networked computer system, determining which of the devices are anticipated to be affected by the anomaly by using pattern correlations across the plurality of hosts, servers, and computer sites, and alerting the devices.

20. The system of claim **19**, wherein the data collection and processing center further determines which of the devices have been affected by the anomaly and alerts the devices.

21. The system of claim **19**, wherein at least one of the plurality of devices comprises a firewall, and the data

collection and processing center further adjusts the firewall of each of the devices that is anticipated to be affected by the anomaly responsive to the detection of the anomaly.

22. The system of claim **19**, wherein the anomaly comprises one of an intrusion, an intrusion attempt, and reconnaissance activity.

23. The system of claim **19**, wherein the data collection and processing center detects the anomaly by analyzing a plurality of data packets with respect to predetermined patterns.

24. The system of claim **23**, wherein the data collection and processing center analyzes data packets that have been received by at least two of the plurality of devices.

25. The system of claim **19**, wherein the data collection and processing center adjusts anomaly detection sensitivity and alarm thresholds based on the detected anomaly.

26. A data collection and processing center comprising a computer with a firewall coupled to a computer network, the data collection and processing center monitoring data communicated to the network, and detecting an anomaly in the network using network-based intrusion detection techniques comprising analyzing data entering into a plurality of hosts, servers, and computer sites in the networked computer system.

27. The data collection and processing center of claim **26**, further comprising determining which of a plurality of devices that are connected to the network are anticipated to be affected by the anomaly by using pattern correlations across the plurality of hosts, servers, and computer sites, and alerting the devices.

28. The data collection and processing center of claim **26**, wherein the data collection and processing center further determines which of a plurality of devices that are connected to the network have been affected by the anomaly and alerts the devices.

29. The data collection and processing center of claim **26**, wherein the data collection and processing center further adjusts a firewall of each of a plurality of devices that is connected to the network that is anticipated to be affected by the anomaly responsive to the detection of the anomaly.

30. The data collection and processing of claim **26**, wherein the anomaly comprises one of an intrusion, an intrusion attempt, and reconnaissance activity.

31. The data collection and processing of claim **26**, wherein the data collection and processing center detects the anomaly by analyzing a plurality of data packets with respect to predetermined patterns.

32. The data collection and processing of claim **31**, wherein the data collection and processing center analyzes data packets that have been received by at least two devices that are connected to the network.

33. The data collection and processing of claim **26**, wherein the data collection and processing center adjusts anomaly detection sensitivity and alarm thresholds based on the detected anomaly.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO.	: 6,715,084 B2	Page 1 of 1
DATED	: March 30, 2004	
INVENTOR(S)	: Jeffrey A. Aaron and Thomas Anschutz	

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 1,
Line 57, "P spoofing" should read -- IP spoofing --

Signed and Sealed this

Eighth Day of June, 2004

Jon W. Dudas

JON W. DUDAS
Acting Director of the United States Patent and Trademark Office

EXHIBIT D

US006314409B2

(12) **United States Patent**
Schneck et al.

(10) **Patent No.: US 6,314,409 B2**
(45) **Date of Patent: *Nov. 6, 2001**

(54) **SYSTEM FOR CONTROLLING ACCESS AND DISTRIBUTION OF DIGITAL PROPERTY**

(75) Inventors: **Paul B. Schneck**, Potomac; **Marshall D. Abrams**, Silver Spring, both of MD (US)

(73) Assignee: **Veridian Information Solutions**

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 10 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **09/178,606**

(22) Filed: **Oct. 26, 1998**

Related U.S. Application Data

(62) Division of application No. 08/968,887, filed on Nov. 5, 1997, which is a continuation of application No. 08/584,493, filed on Jan. 11, 1996, now abandoned.

(51) **Int. Cl.**7 .... **H04L 9/00**

(52) **U.S. Cl.** **705/54**; 705/57; 380/259; 380/287; 380/59; 713/182; 713/189; 713/193; 713/194; 713/200

(58) **Field of Search** 395/186, 187.01, 395/188.01; 380/4, 23, 25, 49, 50, 59, 200–204, 210, 255, 259, 277, 278, 279, 29, 30, 287; 705/43, 51, 52, 54, 57, 58; 713/150–152, 182, 189, 193, 194, 200, 201, 202

(56) **References Cited**

U.S. PATENT DOCUMENTS

3,504,132 3/1970 Wallace, Jr. .
3,648,020 * 3/1972 Tateisi et al. 705/43

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

0332707 9/1989 (EP) .

(List continued on next page.)

OTHER PUBLICATIONS

Weber, R., “Metering Technologies For Digital Intellectual Property,” A Report to the Internatinal Federation of Reproduction Rights Organization, Oct. 1994, pp. 1–29.

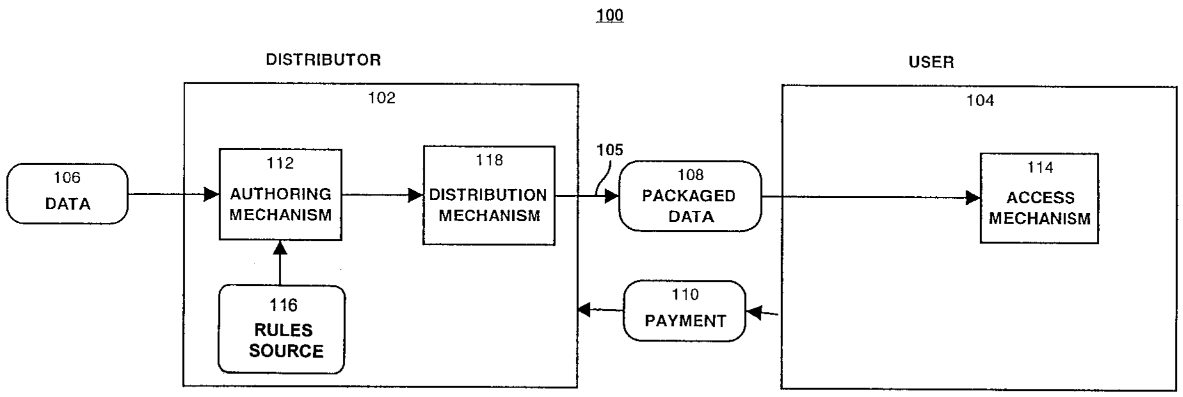
(List continued on next page.)

Primary Examiner—Bernarr E. Gregory
(74) *Attorney, Agent, or Firm*—Pillsbury Winthrop

(57) **ABSTRACT**

A method and device are provided for controlling access to data. Portions of the data are protected and rules concerning access rights to the data are determined. Access to the protected portions of the data is prevented, other than in a non-useable form; and users are provided access to the data only in accordance with the rules as enforced by a mechanism protected by tamper detection. A method is also provided for distributing data for subsequent controlled use of those data. The method includes protecting portions of the data; preventing access to the protected portions of the data other than in a non-useable form; determining rules concerning access rights to the data; protecting the rules; and providing a package including: the protected portions of the data and the protected rules. A user is provided controlled access to the distributed data only in accordance with the rules as enforced by a mechanism protected by tamper protection. A device is provided for controlling access to data having protected data portions and rules concerning access rights to the data. The device includes means for storing the rules; and means for accessing the protected data portions only in accordance with the rules, whereby user access to the protected data portions is permitted only if the rules indicate that the user is allowed to access the portions of the data.

43 Claims, 26 Drawing Sheets



U.S. PATENT DOCUMENTS

3,764,742	10/1973	Abbott et al. .	
3,798,359	3/1974	Feistel .	
3,878,331	4/1975	Morgan et al. .	
3,906,460	9/1975	Halpern .	
3,911,216	10/1975	Bartek et al. .	
3,944,976	3/1976	France .	
3,958,081	5/1976	Ehrsam et al. .	
3,996,449	12/1976	Attanasio et al. .	
4,004,089	1/1977	Richard et al. .	
4,028,678	6/1977	Moran .	
4,037,215	7/1977	Birney et al. .	
4,074,066	2/1978	Ehrsam et al. .	
4,087,856	5/1978	Attanasio .	
4,120,030	10/1978	Johnstone .	
4,168,396	9/1979	Best .	
4,183,085	1/1980	Roberts et al. .	
4,193,131	3/1980	Lennon et al. .	
4,206,315	6/1980	Matyas et al. .	
4,238,854	12/1980	Ehrsam et al. .	
4,246,638	1/1981	Thomas .	
4,264,782	4/1981	Konheim .	
4,278,837	7/1981	Best .	
4,281,215	7/1981	Atalla .	
4,306,289	12/1981	Lumley .	
4,319,079	3/1982	Best .	
4,323,921	4/1982	Guillou .	
4,433,207	2/1984	Best .	
4,446,519	5/1984	Thomas .	
4,454,594	6/1984	Heffron et al. .	
4,458,315	7/1984	Uchenick .	
4,465,901	8/1984	Best .	
4,471,163	9/1984	Donald et al. .	
4,529,870	7/1985	Chaum	235/380
4,558,176	12/1985	Arnold et al. .	
4,646,234	2/1987	Tolman et al.	380/4
4,658,093	4/1987	Hellman	380/25
4,757,533	7/1988	Allen et al.	380/25
4,796,181	1/1989	Wiedemer	380/4 X
4,827,508	5/1989	Shear .	
4,924,378	5/1990	Hershey et al. .	
4,932,054	6/1990	Chou et al.	380/4
4,937,863	6/1990	Robert et al.	380/4
4,953,209	8/1990	Ryder, Sr. et al.	380/23
4,961,142	10/1990	Elliott et al. .	
4,977,594	12/1990	Shear	380/4
5,010,571	4/1991	Katznelson	380/4
5,014,234	5/1991	Edwards, Jr. .	
5,023,907	6/1991	Johnson et al.	380/4
5,027,396	6/1991	Platteter et al. .	
5,047,928	9/1991	Wiedemer .	
5,050,213	9/1991	Shear .	
5,058,162	10/1991	Santon et al. .	
5,058,164	10/1991	Elmer et al.	380/50
5,103,476	4/1992	Waite et al.	380/4
5,113,519	5/1992	Johnson et al. .	
5,146,499	9/1992	Geffrotin	380/23
5,159,182	10/1992	Eisele	235/492
5,191,193	3/1993	LeRoux	235/379
5,204,897	4/1993	Wyman	380/4
5,222,134	6/1993	Waite et al.	380/4
5,235,642	8/1993	Wobber et al. .	
5,247,575	9/1993	Sprague et al. .	
5,260,999	11/1993	Wyman .	
5,263,157	11/1993	Janis .	
5,263,158	11/1993	Janis .	
5,291,596	3/1994	Mita	395/600
5,301,231	4/1994	Abraham et al. .	
5,319,705	6/1994	Halter et al. .	
5,337,357	8/1994	Chou et al. .	
5,339,091	8/1994	Yamazaki et al. .	
5,345,588	9/1994	Greenwood et al. .	
5,347,578	9/1994	Duxbury .	
5,369,702	11/1994	Shanton .	
5,386,469	1/1995	Yearsley et al. .	
5,386,471	1/1995	Bianco .	
5,388,156	2/1995	Blackledge, Jr. et al. .	
5,392,351	2/1995	Hasebe et al. .	
5,394,469	2/1995	Nagel et al. .	
5,400,403	3/1995	Fahn et al. .	
5,410,598	4/1995	Shear .	
5,432,849	7/1995	Johnson et al. .	
5,438,508	8/1995	Wyman .	
5,442,541	8/1995	Hube et al. .	
5,450,489	9/1995	Ostrover et al. .	
5,457,746	10/1995	Dolphin .	
5,473,687	12/1995	Lipscomb et al. .	
5,504,814	4/1996	Miyahara .	
5,530,235	6/1996	Stefik et al. .	
5,574,648	5/1998	Ryan et al. .	
5,592,549	1/1997	Nagel et al.	380/4
5,594,491	1/1997	Hodge et al. .	
5,594,936	1/1997	Rebec et al. .	
5,615,264	3/1997	Kazmierczak et al. .	
5,629,980	5/1997	Stefik et al. .	
5,638,443	6/1997	Stefik et al. .	
5,646,992	7/1997	Subler et al. .	
5,671,276	9/1997	Eyer et al. .	
5,673,316	9/1997	Auerbach et al.	380/4
5,677,953	10/1997	Dolphin .	
5,703,951	12/1997	Dolphin .	
5,754,649	5/1998	Ryan et al. .	
5,787,172	7/1998	Arnold .	
5,796,839	8/1998	Ishiguro .	
5,870,474	2/1999	Wasilewski et al. .	
5,892,900	4/1999	Ginter .	
5,910,987	6/1999	Ginter .	

FOREIGN PATENT DOCUMENTS

2236604	4/1991	(GB) .
9500355-4	8/1996	(SE) .
WO9220022	11/1992	(WO) .
WO93/01550	1/1993	(WO) .
96/27155	9/1996	(WO) .

OTHER PUBLICATIONS

Clark, P.C. and Hoffman, L.J., "Bits: A Smartcard Protected Operating System," Communications of the ACM, Nov. 1994, vol. 37, No. 11, pp. 66–70, and 94.

Saigh, W.K., "Knowledge is Sacred," Video Pocket/Page Reader Systems, Ltd., 1992.

Kahn, R.E., "Deposit, Registration And Recordation In an Electronic Copyright Management System," Corporation for National Research Initiatives, Virginia, Aug. 1992, pp. 1–29.

Hilts, P. Mutter, J., and Taylor, S., "Books While U Wait," Publishers Weekly, Jan. 3. 1994, pp. 48–50.

Strattner, A., "Cash register on a chip" may revolutionize software pricing and distribution; Wave Systems Corp., Computer Shopper. Copyright, Apr. 1994, vol. 14; No.4; p. 62.

O'Conner, M.A., "New distribution option for electronic publishers; iOpener data encryption and metering system for CD–ROM use; Column," CD–ROM Professional Copyright, Mar. 1994, vol. 7; No. 2; p. 134; ISSN:1049–0833.

Willett, S., "Metered PCs:Is your system watching you?"; Wave Systems beta tests new technology, InfoWorld, Copyright, May 2, 1994, p. 84.
Linn, R.J., "Copyright and Information Servicces in the Contest of the National Research and Education Network," IMA Intellectual Property Project Proceedings, Jan. 1994, vol. 1, Issue 1, pp. 9–20.
Perritt, Jr., H.H., "Permissions Headers ad Contract Law," IMA Intellectual Property Project Proceedings, Jan. 1994, vol. 1, Issue 1, pp. 27–48.
Upthegrove, L., and Roberts, R., "Intellectual Property Header Descriptors: A Dynamic Approach," IMA Intellectual Property Project Proceedings, Jan. 1994, vol. 1, Issue 1, pp. 63–66.
Sirbu, M.A., "Internet Billing Service Design and Prototype Implementation, IMA" Intellectual Property Project Proceedings, Jan. 1994, vol. 1, Issue 1, pp. 67–80.
Simmel, S.S., and Godard, I., "Metering and Licensing of Resources: Kala's General Purpose Approach," IMA Intellectual Property Project Proceedings, Jan. 1994, vol. 1, Issue 1, pp. 81–110.
Kahn, R.E., "Deposit, Registration and Recordation in an Electronic Copyright Management System," IMA Intellectual property Project Proceedings, Jan. 1994, vol. 1, Issue 1, pp. 111–120.
Tygar, J.D., and Bennet, Y., "Dyad: A System for Using Physically Secure Coprocessors," IMA Intellectual Property Project Proceedings, Jan. 1994, vol. 1, Issue 1, pp. 121–152.
Griswold, G.N., "A Method for Protecting Copyright on Networks," IMA Intellectual Property Project Proceedings, Jan. 1994, vol. 1, Issue 1, pp. 169–178.
Nelson, T.H., "A Publishing and Royalty Model for Networked Documents," IMA Intellectual Property Project Proceedings, Jan. 1994, vol. 1, Issue 1, pp. 257–259.
European Search Report for Corresponding European Application 95308420.9.
U. Flasche et al., Decentralized Processing of Documents, Comput. & Graphics, vol. 10, No. 2, 1986, pp. 119–131.
R. Mori et al., Superdistribution: The Concept and the Architecture, The Transactions of the IEICE, vol. E 73, No. 7, 1990, Tokyo, JP, pp. 1133–1146.
Rosse, P.E., "Data guard", Forbes, Jun. 6, 1994, pp. 101.
Xiao–Wen Yang et al., Key distribution system for digital video signal, ICSP '96. 1996 3rd International Conference on Signal Processing Proceedings (Cat. No. 96TH8116), vol. 2 1996, pp. 847–50.
E.A.I. Claus, Digital network for video surveillance and video distribution, Proc. SPIE—Int. Soc., Opt. Eng. vol. 2952 1996, pp. 194–204.
R. J. Bankapur et al., Switched digital video access networks, Bell Labs Tech. J. vol. 1 No. 1 Summer 1996, pp. 66–77.
C.A. Mandel et al., Intellectual access to digital documents:joining proven principles with new technologies, Cat. Classif. Q., vol. 22, No. 3–4 1996, pp. 25–42.
B.J. Goldsmith et al., Digital video distribution and transmission, International Broadcasting Convention (Conf. Publ. No. 428) 1996, pp. 26–31.
D. Van Schooneveld, Standardization of conditional access systems for digital pay television, Philips, J. Res. (UK), vol. 50, No. 1–2, 1996, pp. 217–25.
H.D. Wactlar, Intelligent access to digital video: Informedia projectComputer, vol. 29, No. 5, May 1996, pp. 46–52.
J.E. Dail et al., Adaptive digital access protocol: A MAC protocol for multiservice broadband access networks INS, IEEE Commun. Mag. vol. 34, No. 3, Mar. 1996, pp. 104–12.

S. Stevens et al., Informedia: improving access to digital video—Ins, Interactions, vol. 1, No. 4, Oct. 1994, pp. 67–71.

B. Hein et al., RACE 1051: a multigigabit transport and distribution technology for provision of digital video services—INS, Proc. SPIE—Int. Soc. Opt. Eng., vol. 1974, 1993, pp. 26–33.

Chen Ching–Chin et al., Analog, digital and multimedia: implications for information access INS, Online Information 91. 15th International Online Information Meeting Proceedings, 1991, pp. 283–92.

Abrams, M.D. et al, "Cryptography", Information Security—An Integrated Collection of Essays, Abrams M.D. et al eds., IEEE Computer Society Press 1995, pp. 350–384.

Choudhury, A.K. et al, "copyright Protection for electronic Publishing Over Computer Networks", IEEE Network, May/Jun. 1995, pp. 12–20.

Ciciora, W.S., "Inside the Set–Top Box", IEEE Spectrum, Apr. 1995, vol. 32, No. 4, pp. 70–75.

Department of Defense Standard, Department of Defense Trusted Computer System Evaluation Criteria, DOD 2500.28–STD,GPO 1986–623.

Marshall Abrams, et al, Generalized Framework For Access Control, Towards Prototyping the ORGCON Policy, Oct. 1991, pp. 1–20, Proc 1991 Nat'l Computer Security Conf.

Marshall D. Abrams, et al, Mediation and Separation in Contemporary Information Technology Systems, 1992, pp. 1–15, Proc. 1991 Nat'l Compute Security Conf.

Marshall D. Abrams, et al, Generalized Framework for Access Control: A Formal Rule Set for The ORGCON Policy, MITRE, Apr. 1992, pp. 1–58.

Marshall D. Abrams, Renewed Unserstanding of Access Control Policies, 1993, pp. 1–10, Proc. 16th National Computer Security Conference.

Leonard J. LaPadula, A Rule–Set Approach to Formal Modeling of a Trusted Computer System, Computing Systems Journal, Winter 1994, vol. 7, No. 1, pp. 113–167, pp. 1–38.

Graubart, R., "On the Need for a third form of Access control", Proceedings of the 12th National Computer Security Conference, 1989, pp. 296–303.

K. Brunnstein and P. P. Sint, eds., KnowRight'95, Intellectual Property Rights and New Technologies: Proceedings of the KnowRigt'95 conference, austrian Computer.

Low, S.H. et al, "Document Marking and Identification using both Line and Word shifting", 1995 INFOCOM Proceedings, IEEE, 1995, pp. 853–860.

McCollum, C.J. et al, "beyond the Pale of MAC and DAC:Defining New Forms of Access Control", Proceedings of the Symposium on Research in Security and Privacy, IEEE Computer Society Press, 1990, pp. 190–200.

National Institute of Standards and technology (NIST) and National Security Agency (NSA), Federal Criteria for Information Technology Security: vol. 1, Protection Profile Development, vol. 11, Registry of Protection Profiles, Version 1.0, Dec. 1992.
Samuelson, P., "Copyright and Digital Libraries", Communications of the AMC, Apr. 1995, vol. 38, No. 3, pp. 15–20 & 110.
Samuelson, P. et al, "A Manifesto Concerning the Legal Protection of Computer Programs", Columbia Law Review, vol. 94, No. 8, pp. 2308–2431.
Sandhu, R.S. "The Typed Access Matrix Model", Proceedings of the Symposium on Research in Security and Provacy, IEEE Computer Society, 1992, pp. 122–136.

Sandhu, R.S. et al., "Implementation Considerations for the Typed Access Marix Model in a Distributed Environment", Proceedings of the 15th National Computer Security Conference, 1992b, pp. 221–235.

Yee, B., "Using Secure Coprocessors", Carnegie Mellon University, School of Computer Science, CMU–CS–94–149, May 1994, (also available Defense Technical Information Center as AD–A281 255).

Maxem Chuk, N.F., Sep. 1994, "Electronic Document Distribution," AT&T technical Journal, pp. 73–80.

* cited by examiner

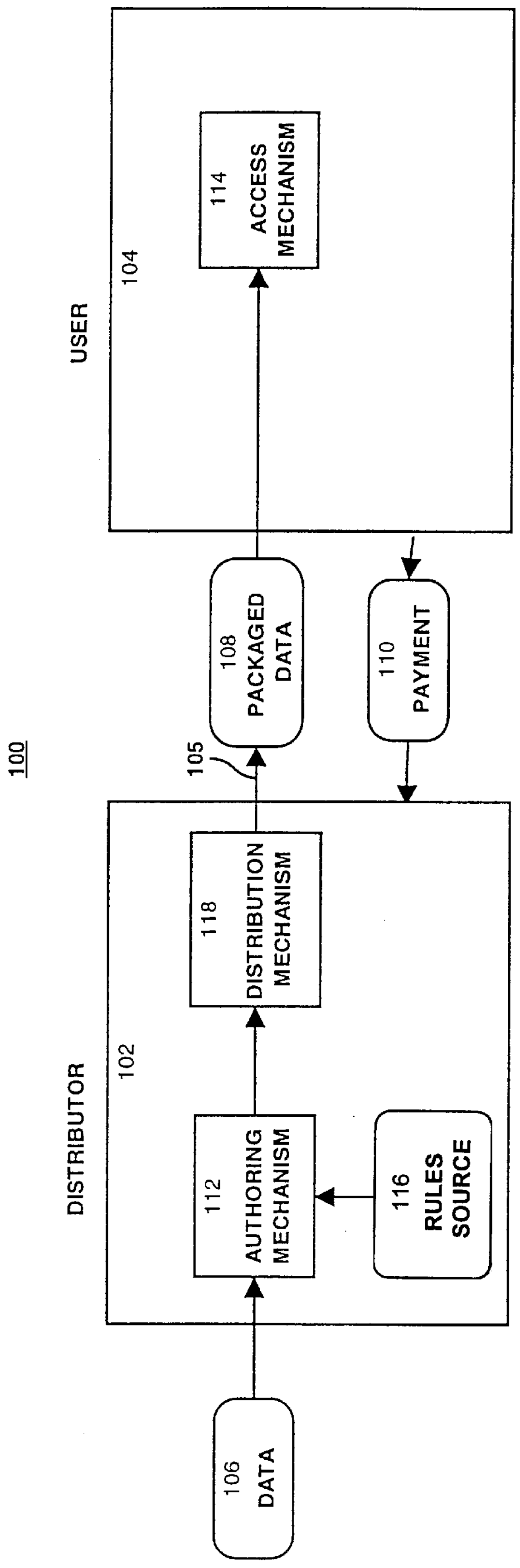

100
DISTRIBUTOR
102
106 DATA
112 AUTHORING MECHANISM
116 RULES SOURCE
118 DISTRIBUTION MECHANISM
105
108 PACKAGED DATA
110 PAYMENT
USER
104
114 ACCESS MECHANISM

FIG. 1

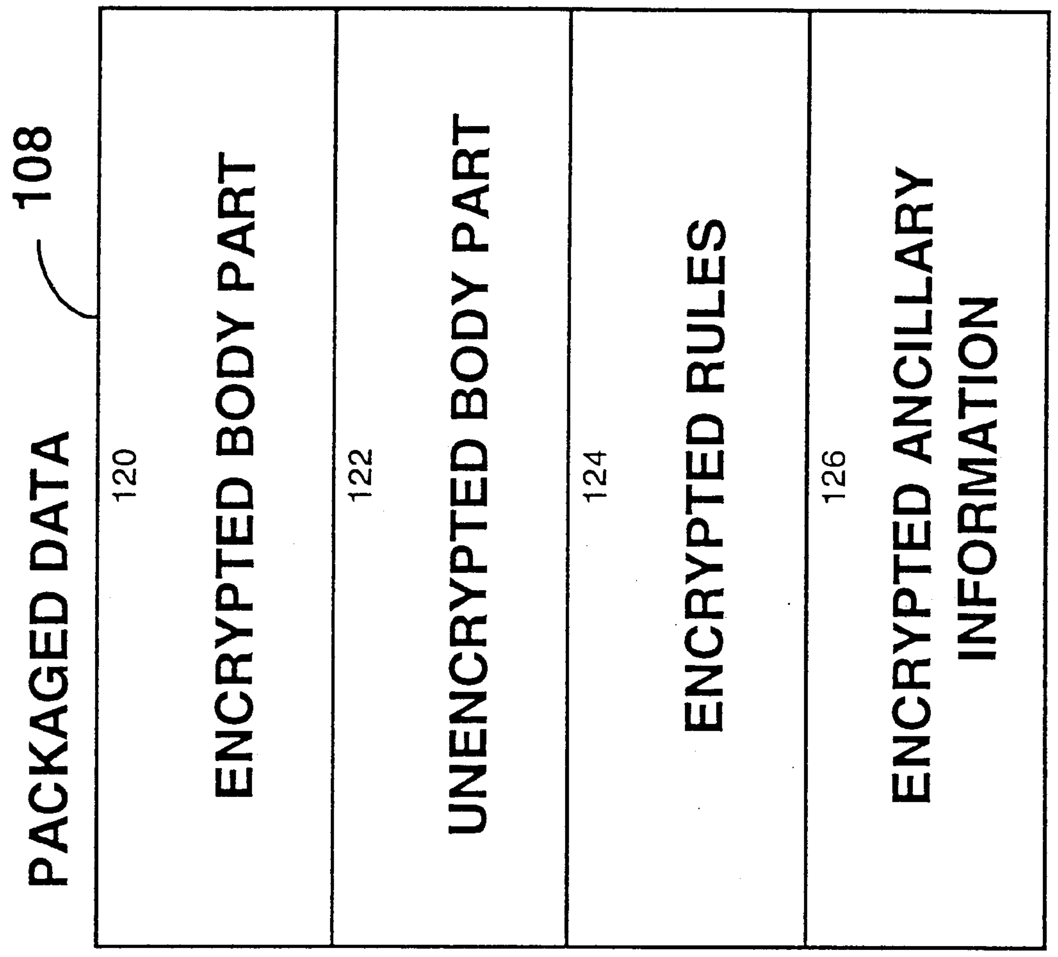

PACKAGED DATA
108
120
ENCRYPTED BODY PART
122
UNENCRYPTED BODY PART
124
ENCRYPTED RULES
126
ENCRYPTED ANCILLARY INFORMATION

FIG. 2

Fig. 3

116

Version number 127
Authentication (hash) 128
License number of these rules 130
Intellectual property identifier 132
First valid generation of the product 134
Last valid generation of the product 136
Encrypted data key 138
Standard permissions 140
Extended permissions 142
Custom permissions 144
Co-requisite rules (permissions) for source data 145
Token/biometrics 146
System IDs/Public keys 147

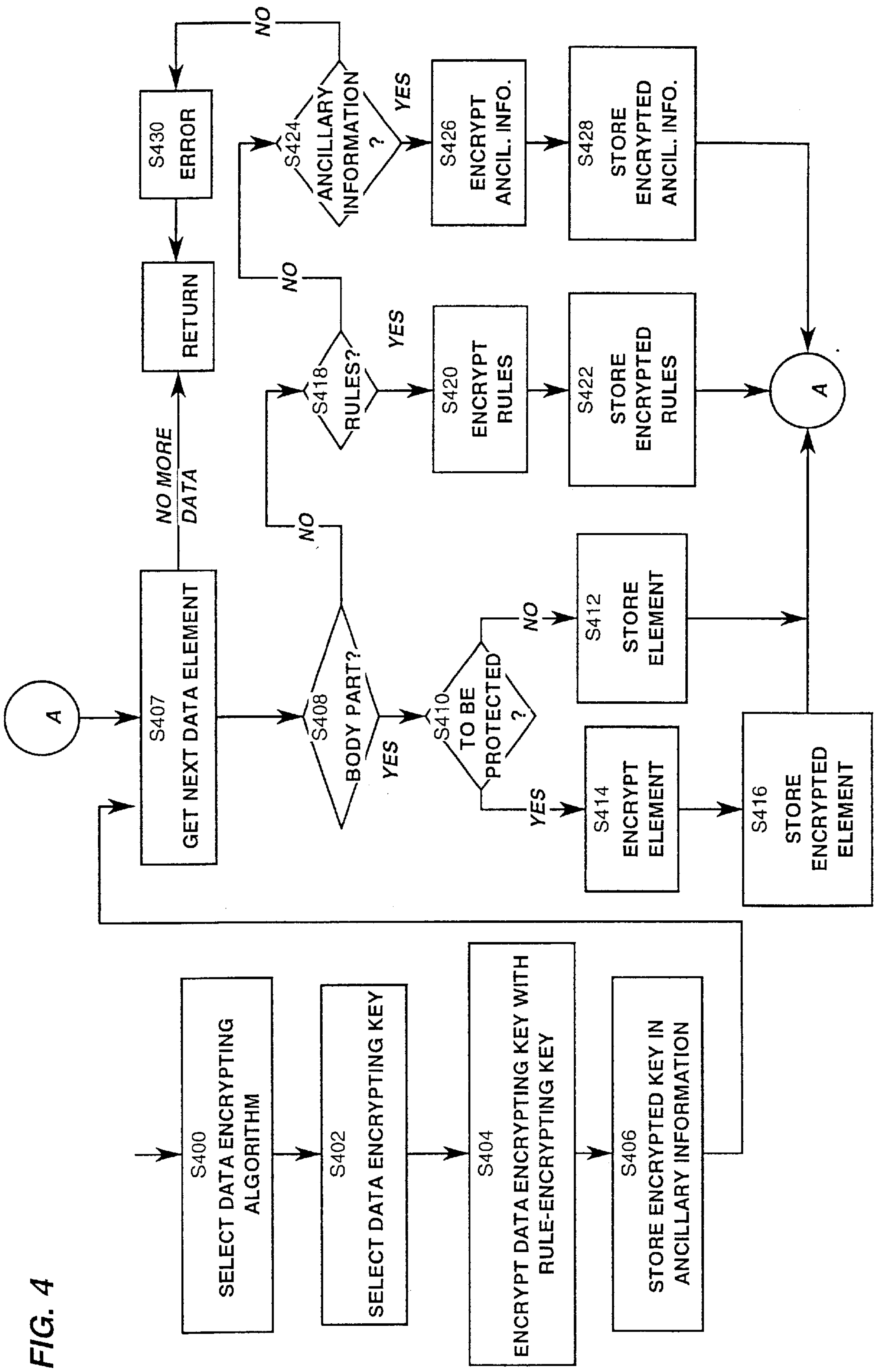
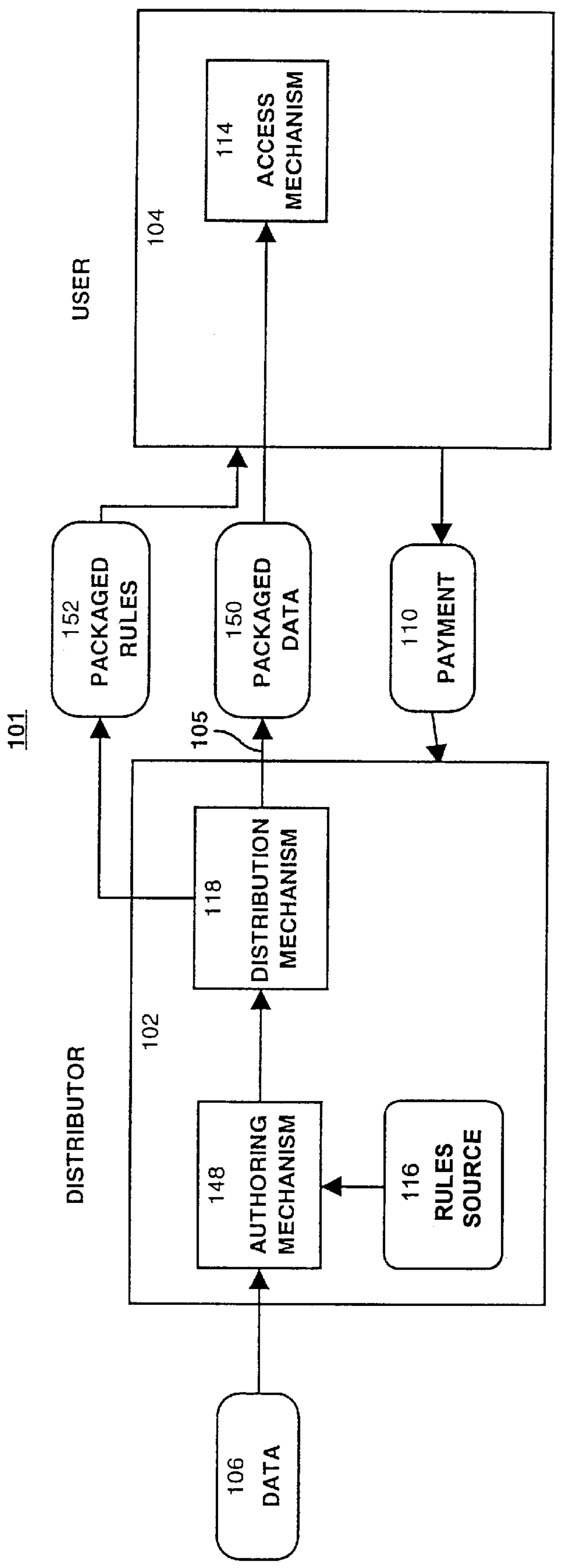

S400 SELECT DATA ENCRYPTING ALGORITHM
S402 SELECT DATA ENCRYPTING KEY
S404 ENCRYPT DATA ENCRYPTING KEY WITH RULE-ENCRYPTING KEY
S406 STORE ENCRYPTED KEY IN ANCILLARY INFORMATION
A
S407 GET NEXT DATA ELEMENT
NO MORE DATA
RETURN
S430 ERROR
S408 BODY PART?
YES
NO
S410 TO BE PROTECTED ?
YES
NO
S412 STORE ELEMENT
S414 ENCRYPT ELEMENT
S416 STORE ENCRYPTED ELEMENT
S418 RULES?
YES
NO
S420 ENCRYPT RULES
S422 STORE ENCRYPTED RULES
S424 ANCILLARY INFORMATION ?
YES
NO
S426 ENCRYPT ANCIL. INFO.
S428 STORE ENCRYPTED ANCIL. INFO.
A

FIG. 4

FIG. 5


USER
104
114 ACCESS MECHANISM
101
152 PACKAGED RULES
105
150 PACKAGED DATA
110 PAYMENT
DISTRIBUTOR
102
118 DISTRIBUTION MECHANISM
148 AUTHORING MECHANISM
116 RULES SOURCE
106 DATA

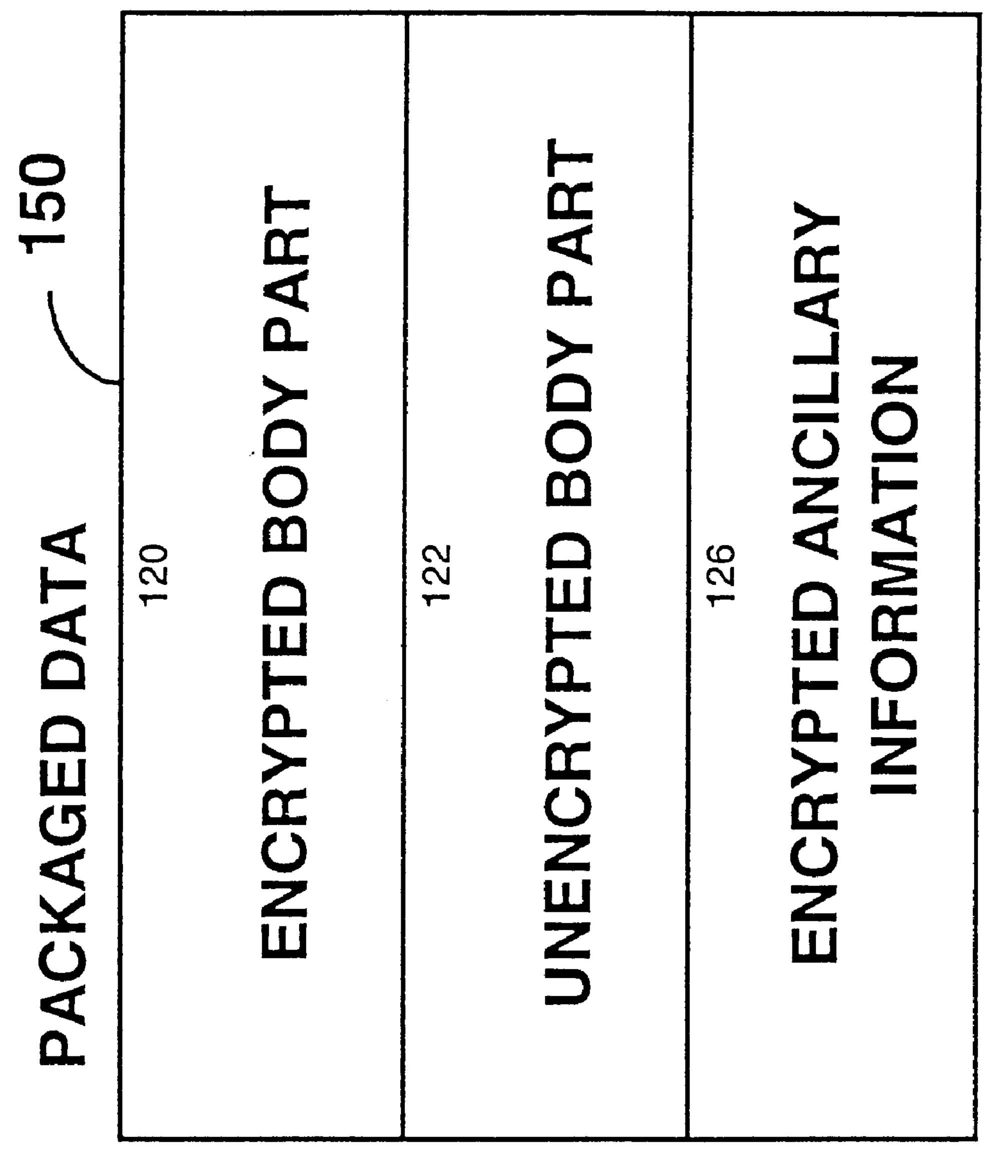
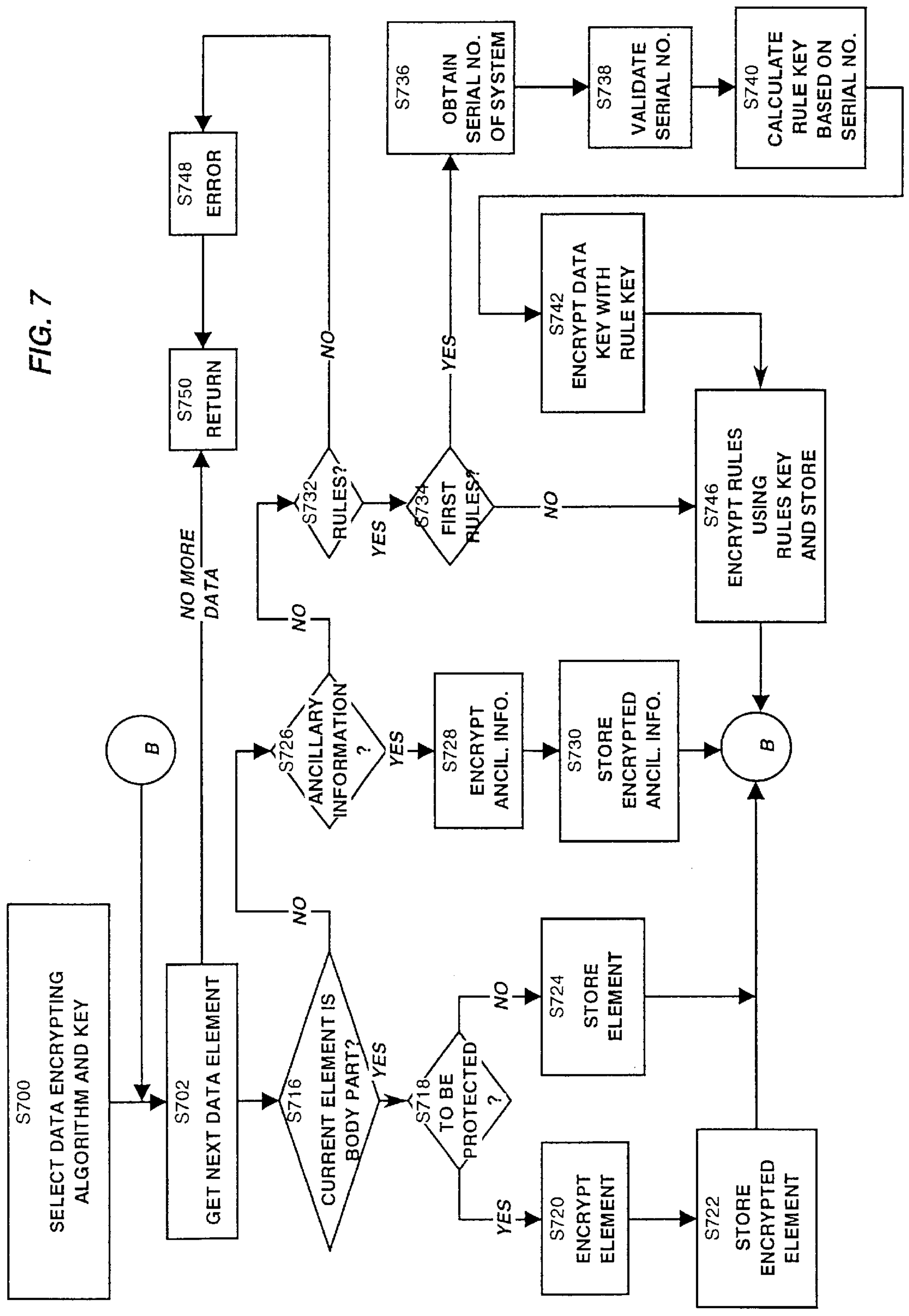

150
PACKAGED DATA
120
ENCRYPTED BODY PART
122
UNENCRYPTED BODY PART
126
ENCRYPTED ANCILLARY INFORMATION

FIG. 6

FIG. 7


S700 SELECT DATA ENCRYPTING ALGORITHM AND KEY
B
S702 GET NEXT DATA ELEMENT
NO MORE DATA
S750 RETURN
S748 ERROR
S716 CURRENT ELEMENT IS BODY PART?
YES
NO
S718 TO BE PROTECTED ?
YES
NO
S720 ENCRYPT ELEMENT
S722 STORE ENCRYPTED ELEMENT
S724 STORE ELEMENT
S726 ANCILLARY INFORMATION ?
YES
NO
S728 ENCRYPT ANCIL. INFO.
S730 STORE ENCRYPTED ANCIL. INFO.
S732 RULES?
YES
NO
S734 FIRST RULES?
YES
NO
S736 OBTAIN SERIAL NO. OF SYSTEM
S738 VALIDATE SERIAL NO.
S740 CALCULATE RULE KEY BASED ON SERIAL NO.
S742 ENCRYPT DATA KEY WITH RULE KEY
S746 ENCRYPT RULES USING RULES KEY AND STORE
B

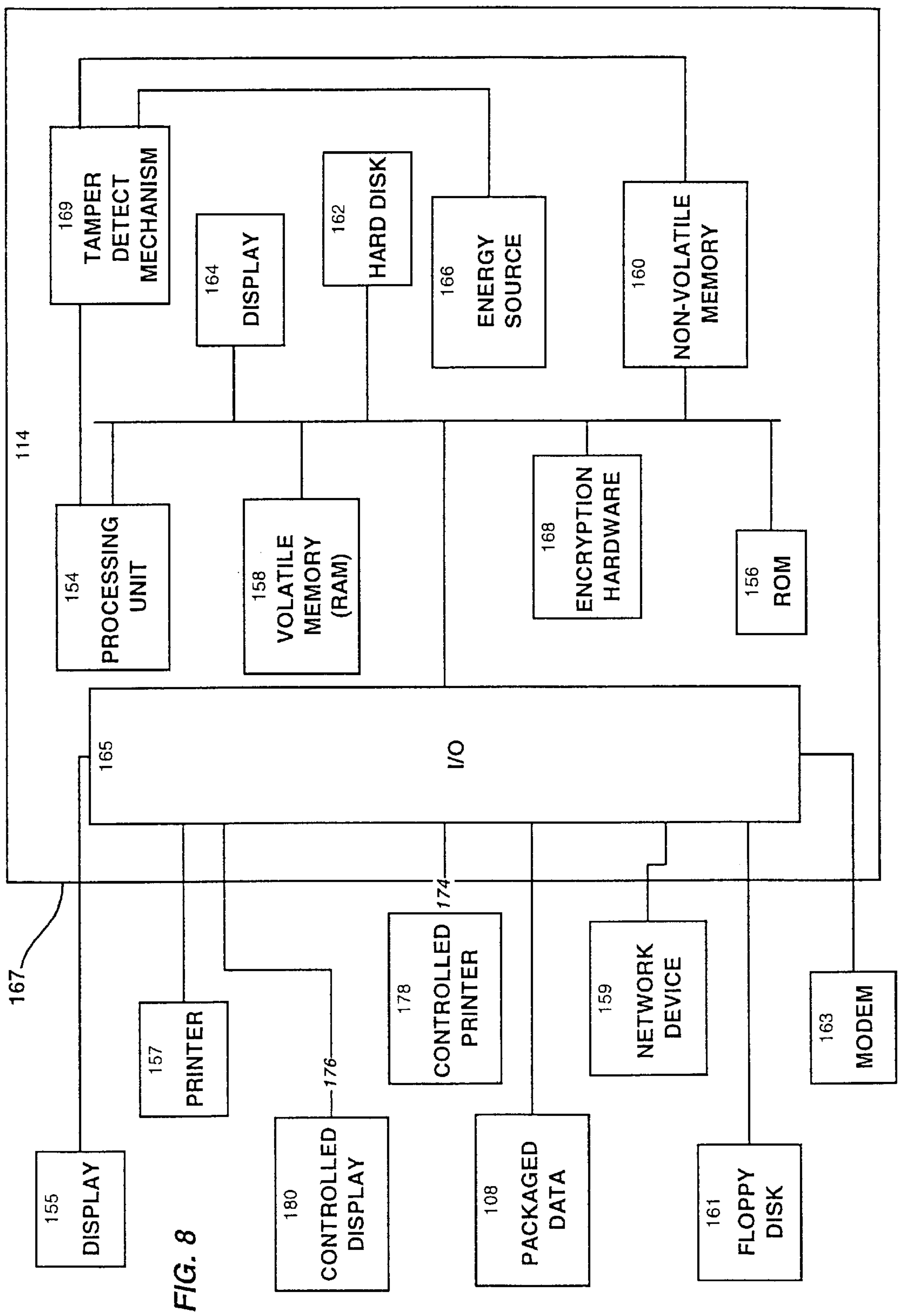

114
169 TAMPER DETECT MECHANISM
164 DISPLAY
162 HARD DISK
166 ENERGY SOURCE
160 NON-VOLATILE MEMORY
154 PROCESSING UNIT
158 VOLATILE MEMORY (RAM)
168 ENCRYPTION HARDWARE
156 ROM
165
I/O
167
174
176
155 DISPLAY
157 PRINTER
180 CONTROLLED DISPLAY
178 CONTROLLED PRINTER
108 PACKAGED DATA
159 NETWORK DEVICE
161 FLOPPY DISK
163 MODEM

FIG. 8

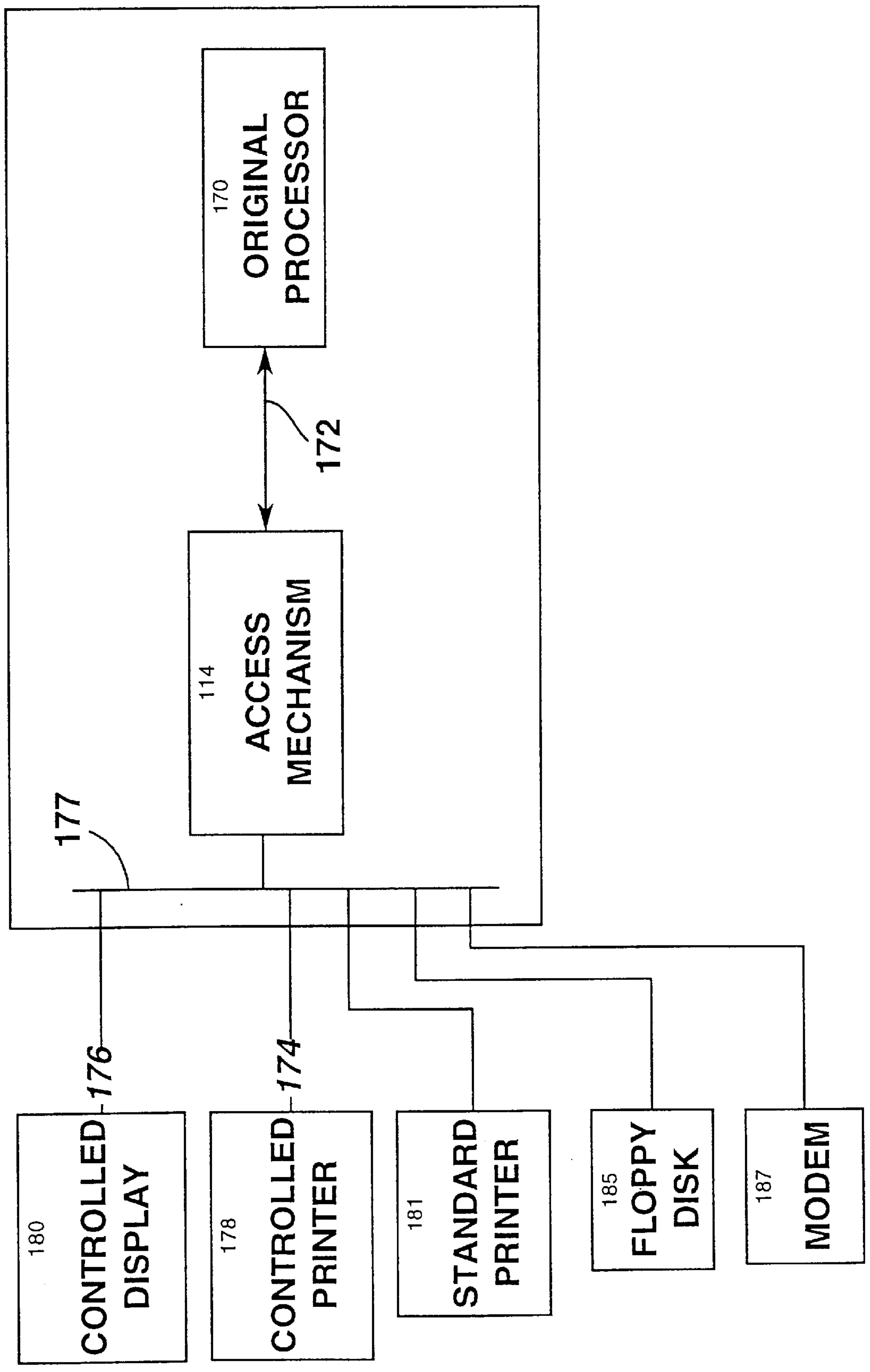

170
ORIGINAL PROCESSOR
172
114
ACCESS MECHANISM
177
176
174
180
CONTROLLED DISPLAY
178
CONTROLLED PRINTER
181
STANDARD PRINTER
185
FLOPPY DISK
187
MODEM

FIG. 9

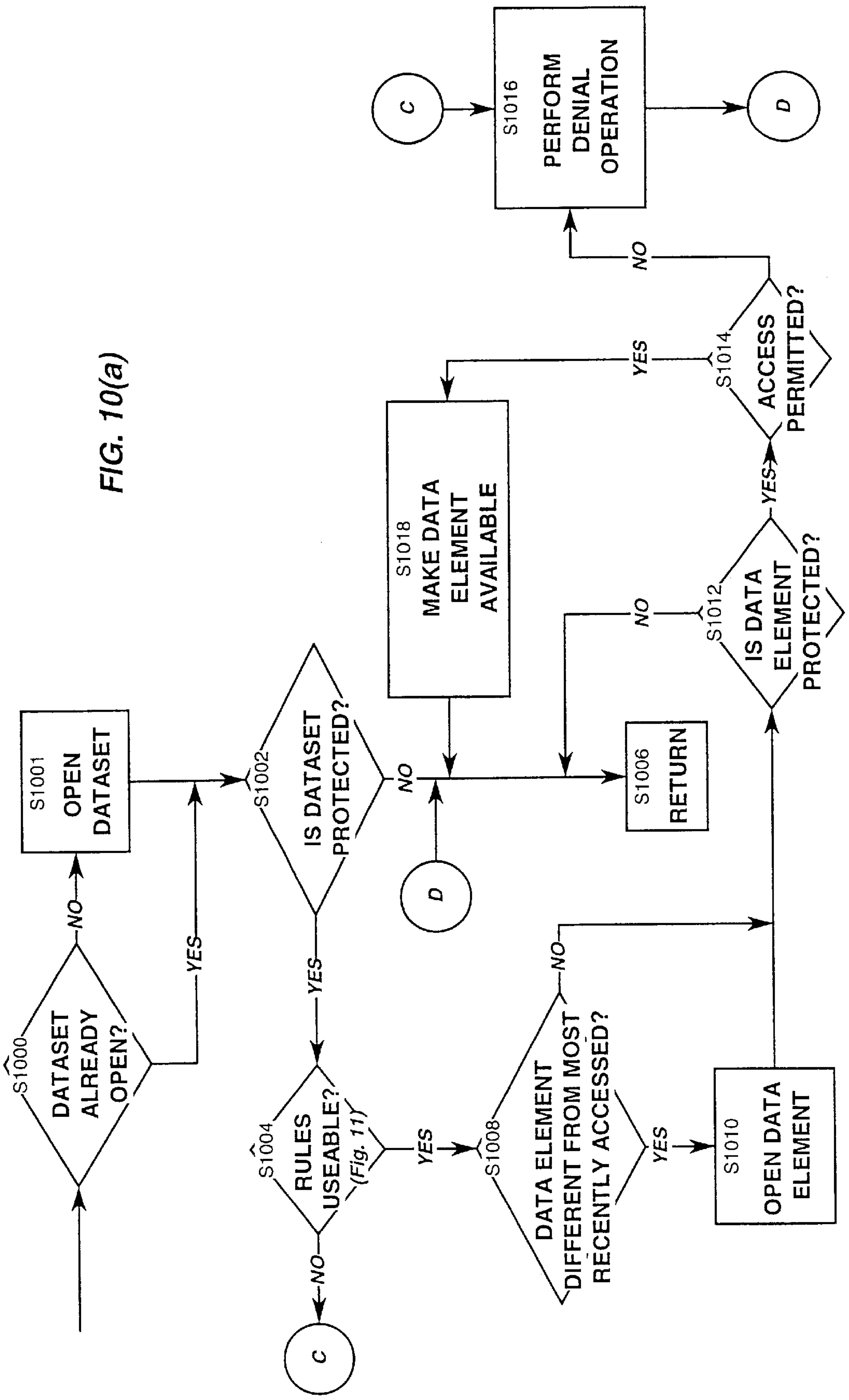
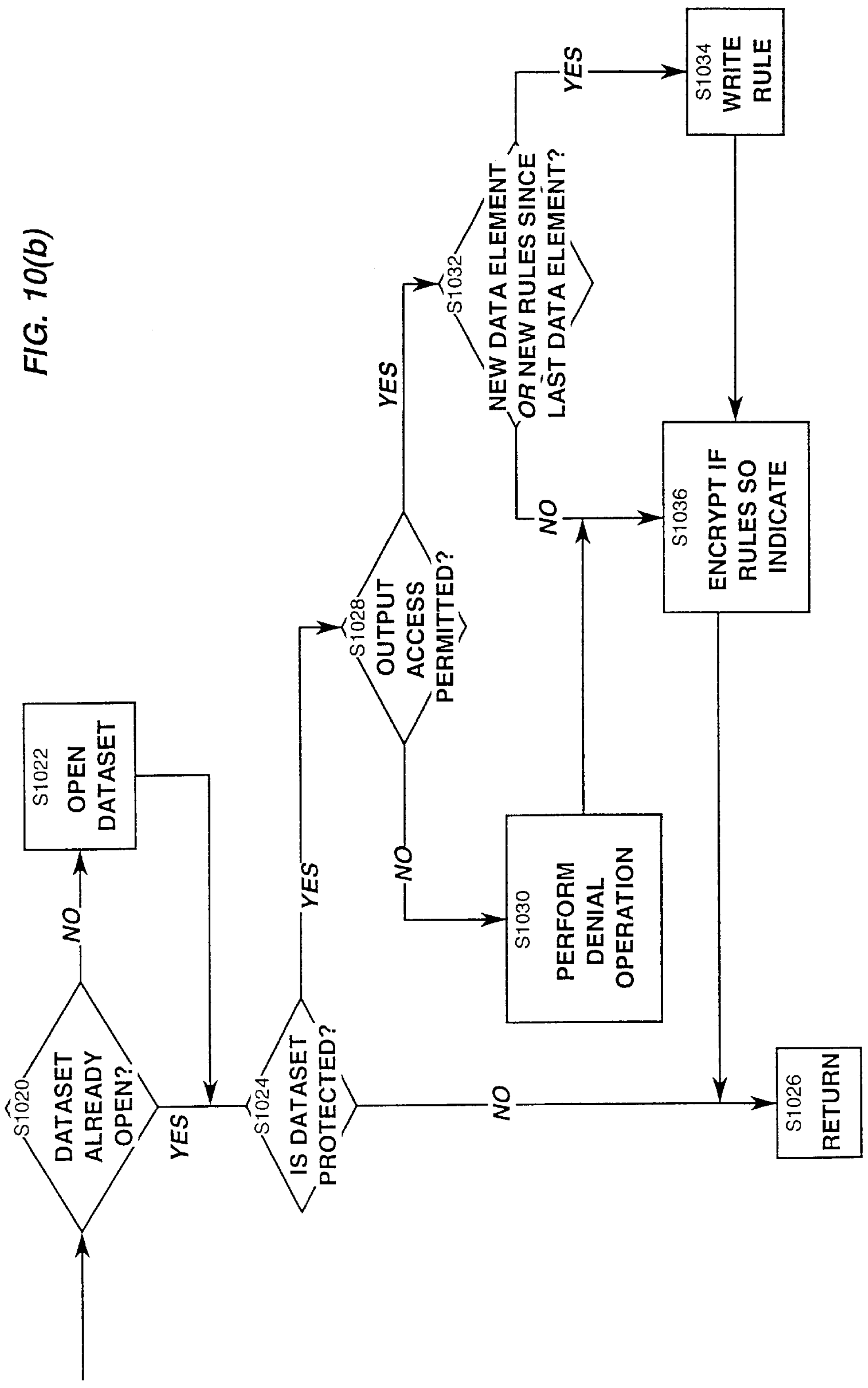

S1000
DATASET ALREADY OPEN?
NO
YES
S1001
OPEN DATASET
S1002
IS DATASET PROTECTED?
NO
YES
S1004
RULES USEABLE? (Fig. 11)
NO
YES
C
D
S1006
RETURN
S1008
DATA ELEMENT DIFFERENT FROM MOST RECENTLY ACCESSED?
NO
YES
S1010
OPEN DATA ELEMENT
S1012
IS DATA ELEMENT PROTECTED?
NO
YES
S1014
ACCESS PERMITTED?
YES
NO
S1018
MAKE DATA ELEMENT AVAILABLE
C
S1016
PERFORM DENIAL OPERATION
D

FIG. 10(a)

FIG. 10(b)


S1020
DATASET ALREADY OPEN?
NO
S1022
OPEN DATASET
YES
S1024
IS DATASET PROTECTED?
NO
YES
S1028
OUTPUT ACCESS PERMITTED?
NO
S1030
PERFORM DENIAL OPERATION
YES
S1032
NEW DATA ELEMENT OR NEW RULES SINCE LAST DATA ELEMENT?
YES
S1034
WRITE RULE
NO
S1036
ENCRYPT IF RULES SO INDICATE
S1026
RETURN

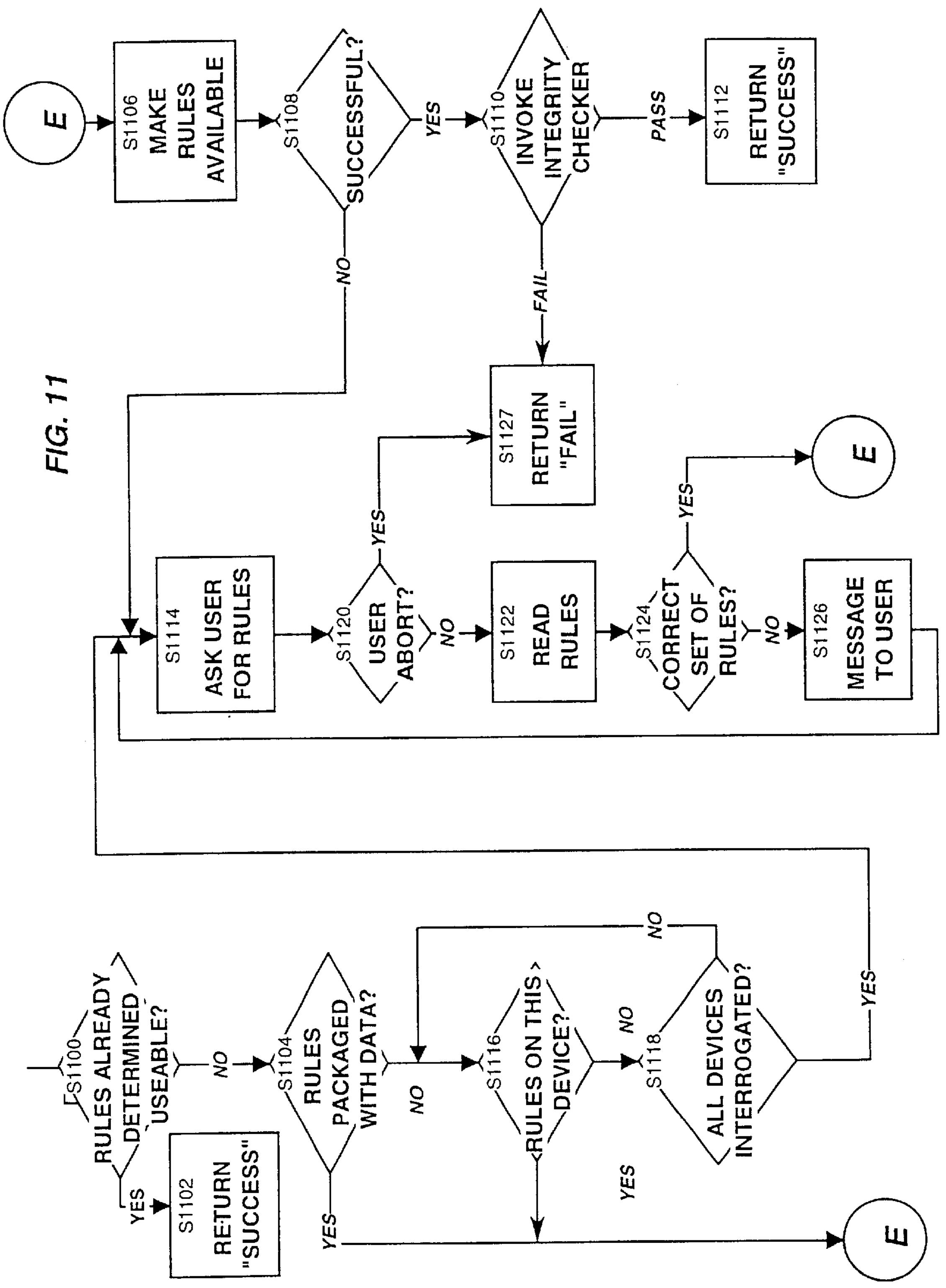

S1100
RULES ALREADY DETERMINED USEABLE?
YES
NO
S1102
RETURN "SUCCESS"
S1104
RULES PACKAGED WITH DATA?
YES
NO
S1116
RULES ON THIS DEVICE?
YES
NO
S1118
ALL DEVICES INTERROGATED?
YES
NO
E
S1114
ASK USER FOR RULES
S1120
USER ABORT?
YES
NO
S1127
RETURN "FAIL"
S1122
READ RULES
S1124
CORRECT SET OF RULES?
YES
NO
E
S1126
MESSAGE TO USER
E
S1106
MAKE RULES AVAILABLE
S1108
SUCCESSFUL?
YES
NO
S1110
INVOKE INTEGRITY CHECKER
PASS
FAIL
S1112
RETURN "SUCCESS"

FIG. 11

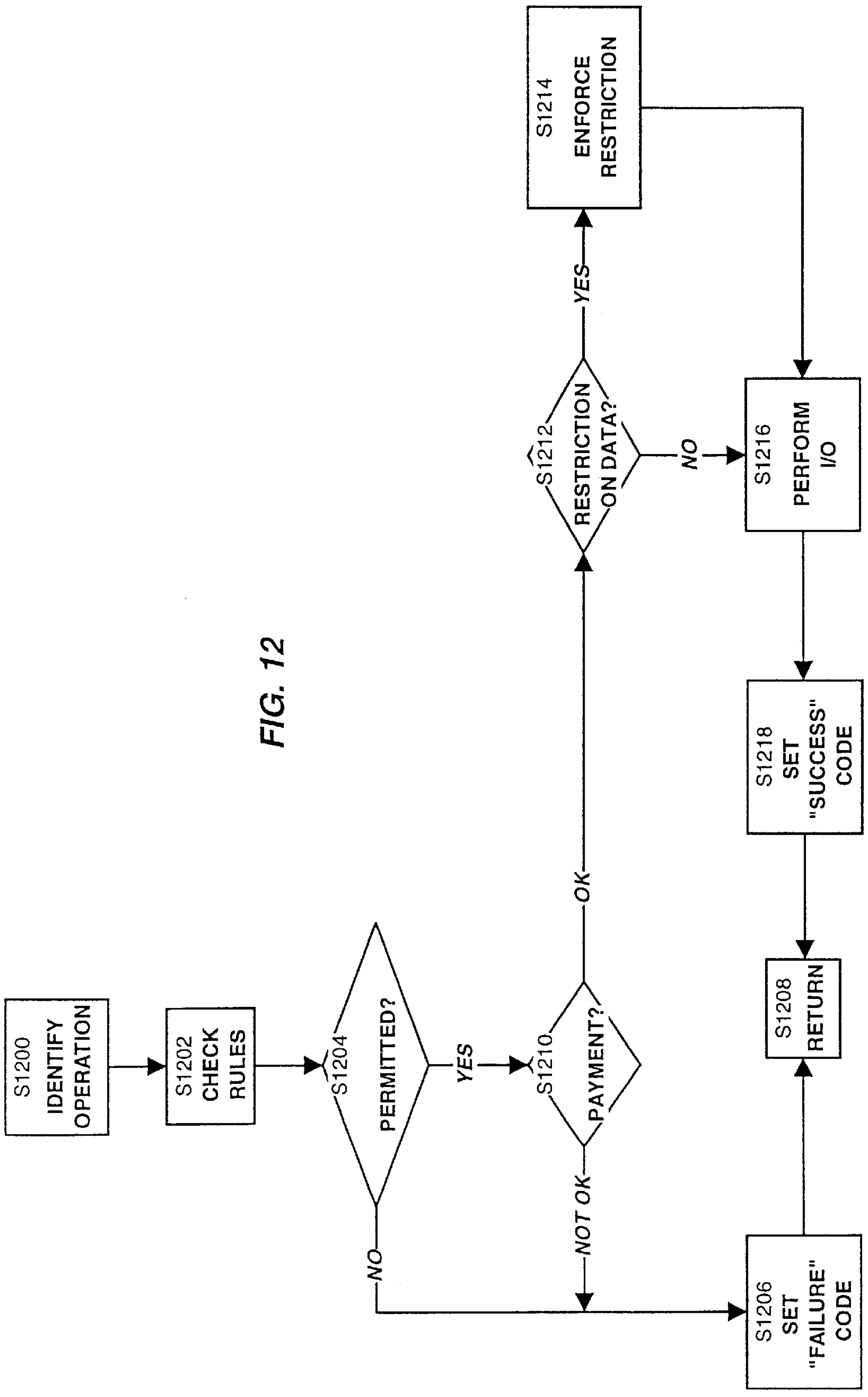

S1200 IDENTIFY OPERATION
S1202 CHECK RULES
S1204 PERMITTED?
YES
NO
S1210 PAYMENT?
OK
NOT OK
S1212 RESTRICTION ON DATA?
YES
NO
S1214 ENFORCE RESTRICTION
S1216 PERFORM I/O
S1218 SET "SUCCESS" CODE
S1206 SET "FAILURE" CODE
S1208 RETURN

FIG. 12

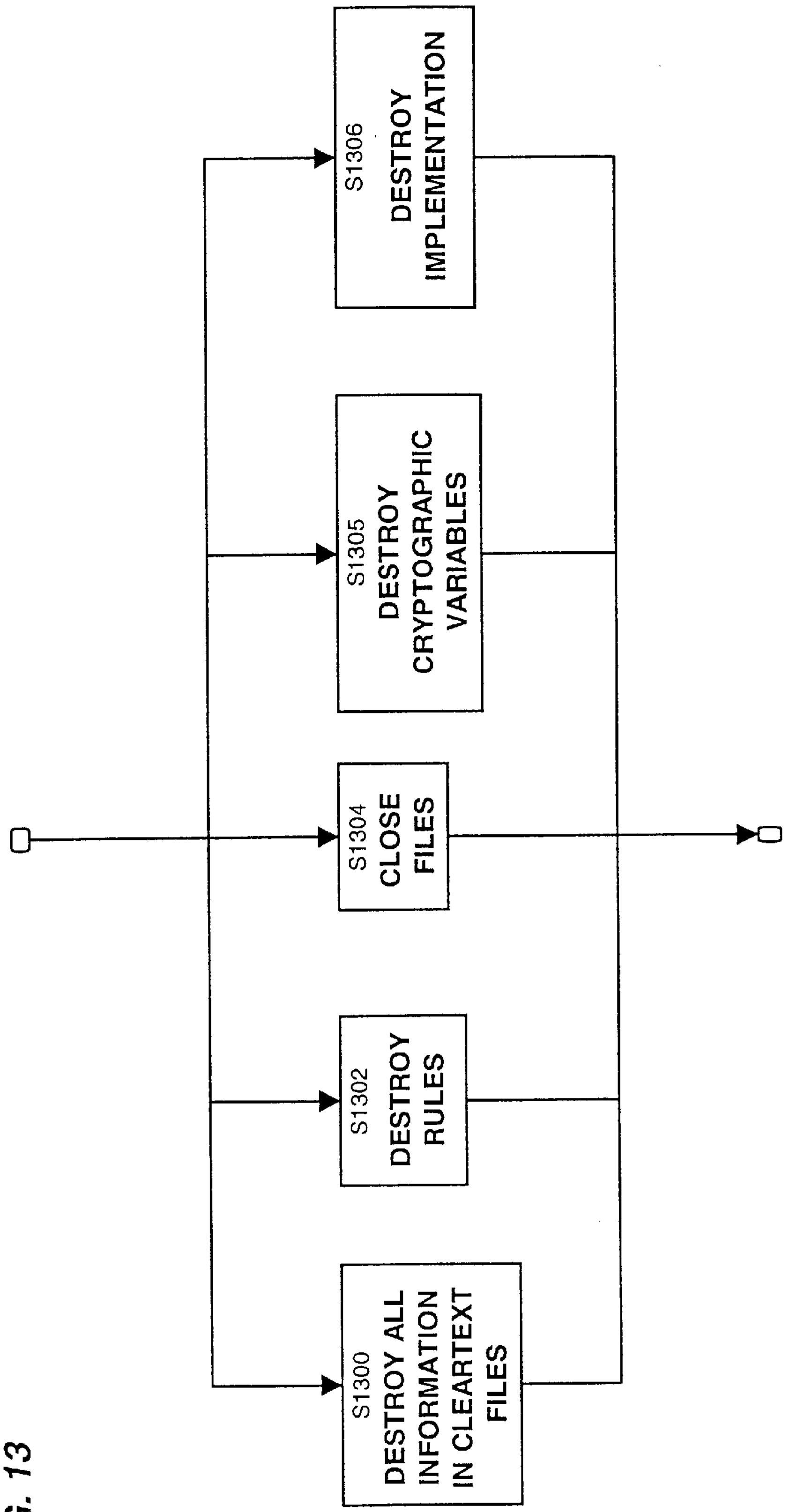

S1300 DESTROY ALL INFORMATION IN CLEARTEXT FILES
S1302 DESTROY RULES
S1304 CLOSE FILES
S1305 DESTROY CRYPTOGRAPHIC VARIABLES
S1306 DESTROY IMPLEMENTATION

FIG. 13

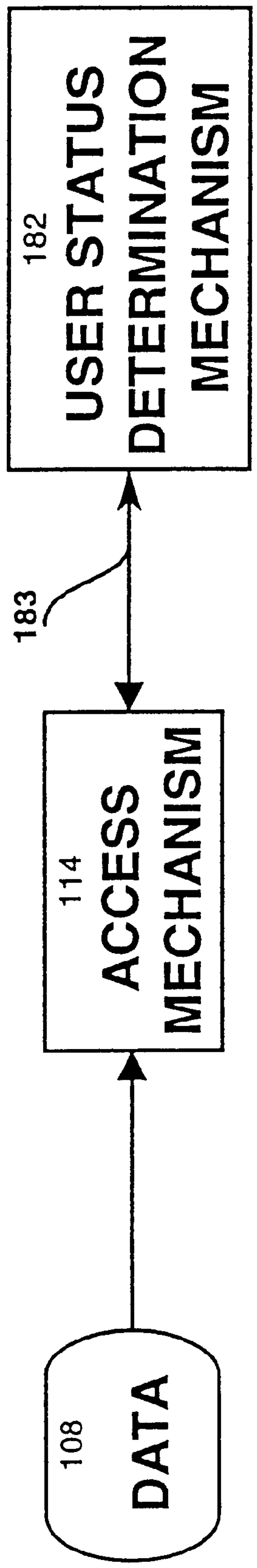
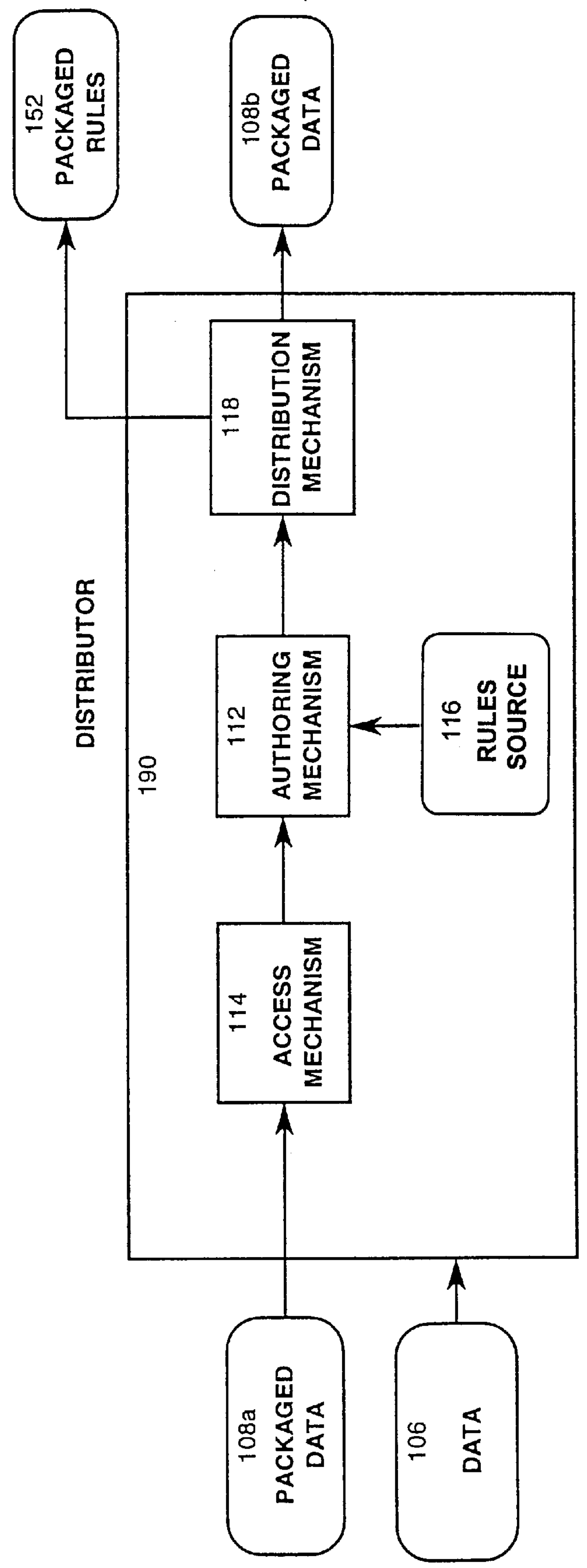

182
USER STATUS
DETERMINATION
MECHANISM
183
114
ACCESS
MECHANISM
108
DATA

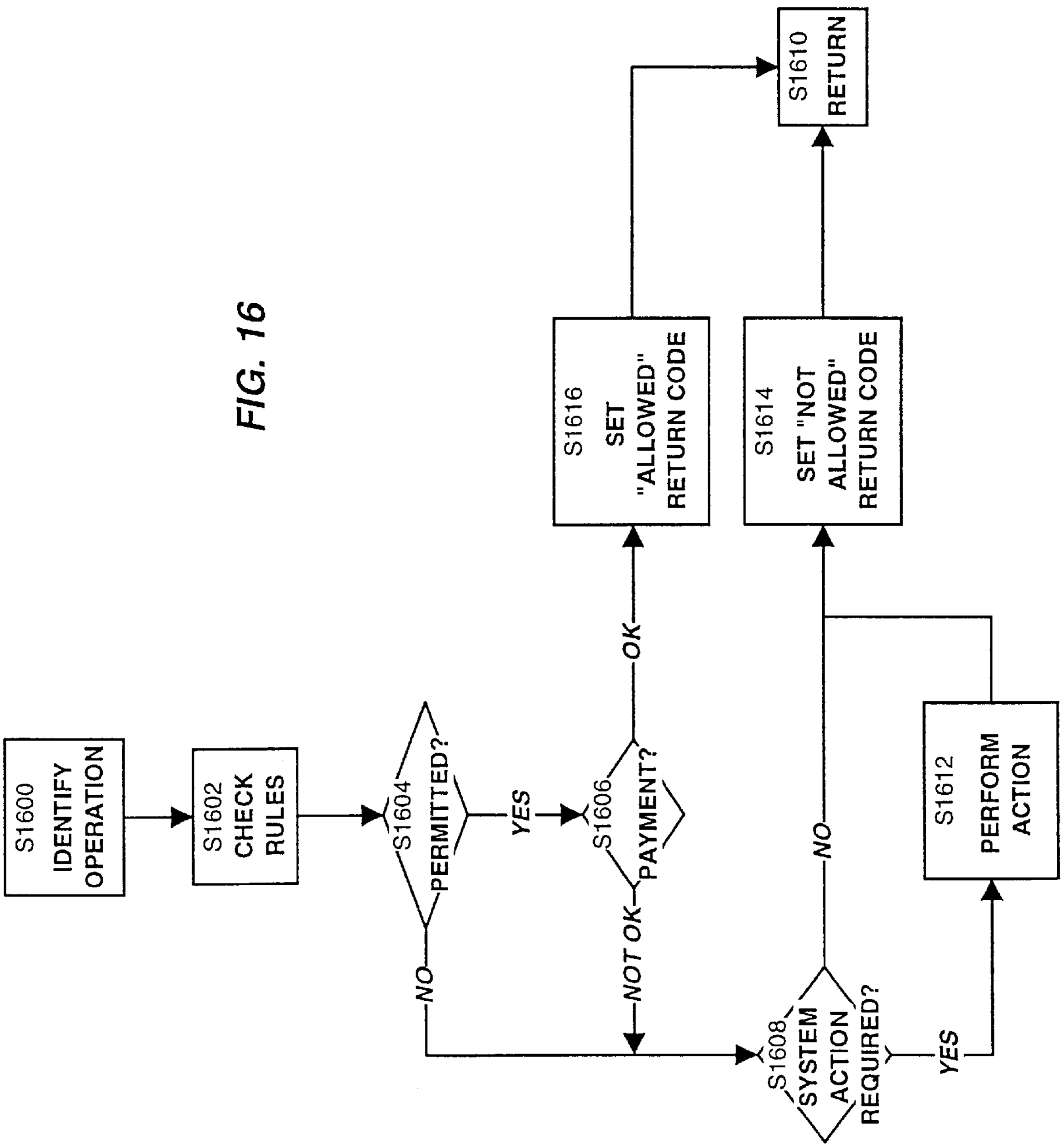
FIG. 14



152 PACKAGED RULES
108b PACKAGED DATA
DISTRIBUTOR
190
118 DISTRIBUTION MECHANISM
112 AUTHORING MECHANISM
116 RULES SOURCE
114 ACCESS MECHANISM
108a PACKAGED DATA
106 DATA

FIG. 15

FIG. 16


S1600 IDENTIFY OPERATION
S1602 CHECK RULES
S1604 PERMITTED?
YES
NO
S1606 PAYMENT?
OK
NOT OK
S1608 SYSTEM ACTION REQUIRED?
YES
NO
S1612 PERFORM ACTION
S1614 SET "NOT ALLOWED" RETURN CODE
S1616 SET "ALLOWED" RETURN CODE
S1610 RETURN

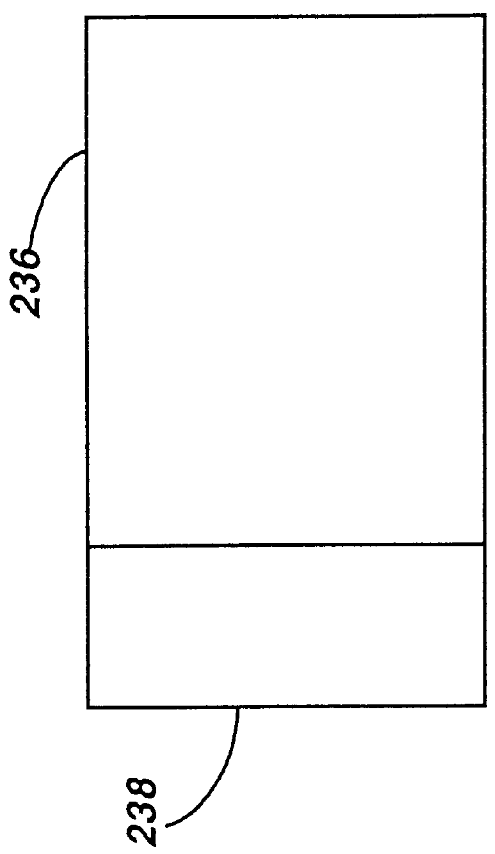


FIG. 17(a)

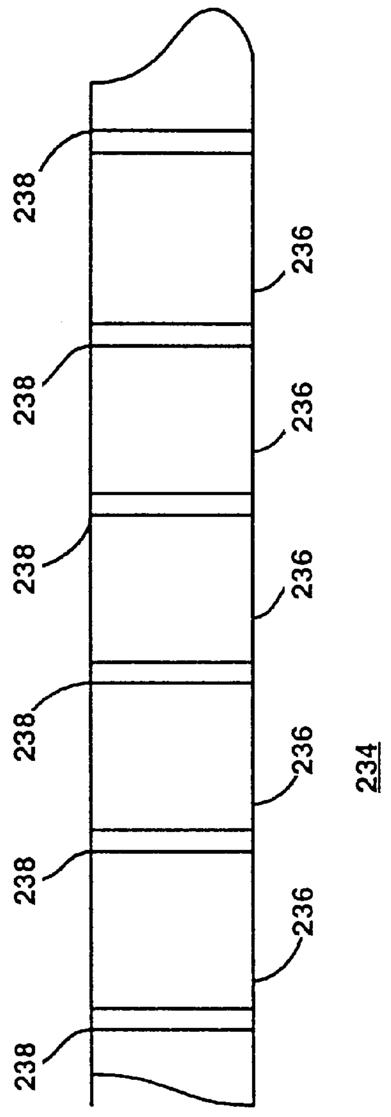


FIG. 17(b)

FIG. 18(b)

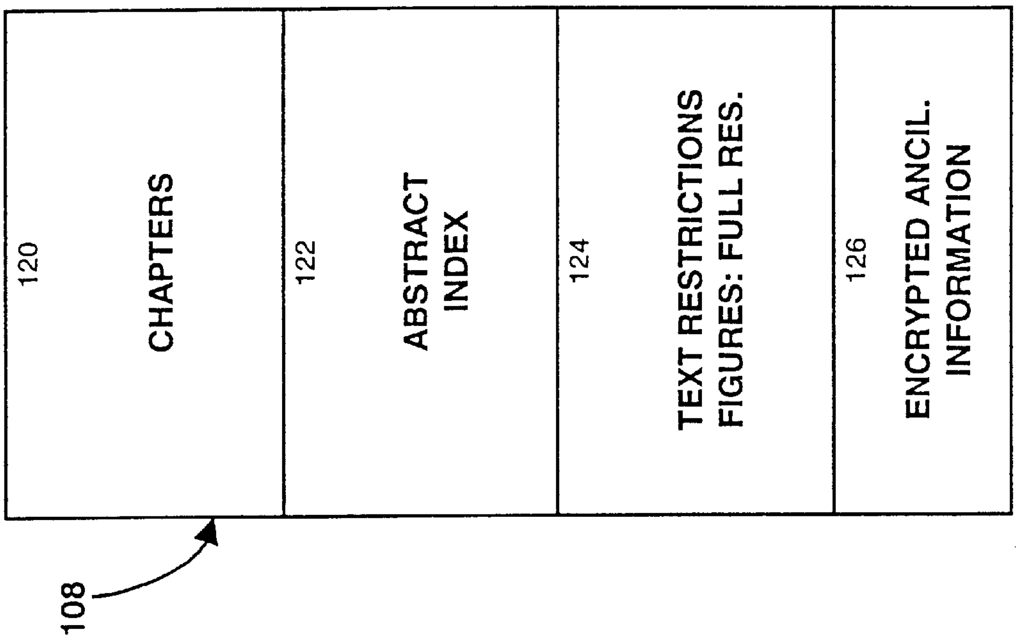
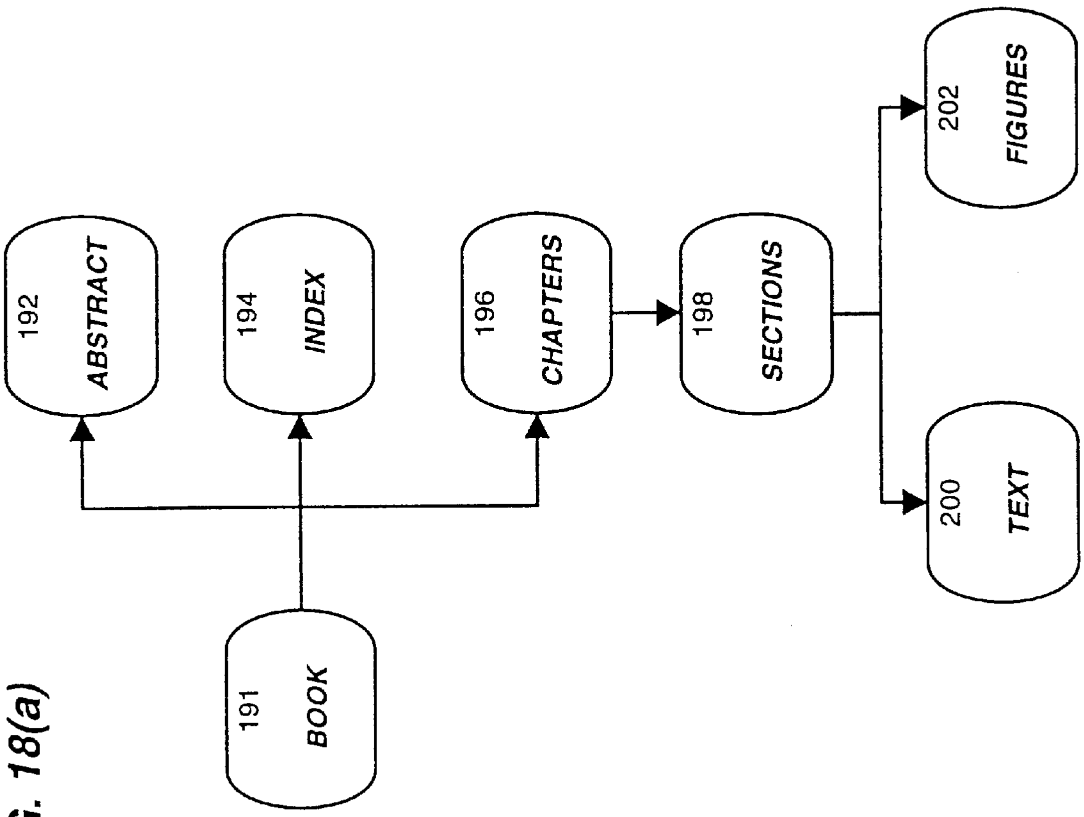


FIG. 18(a)



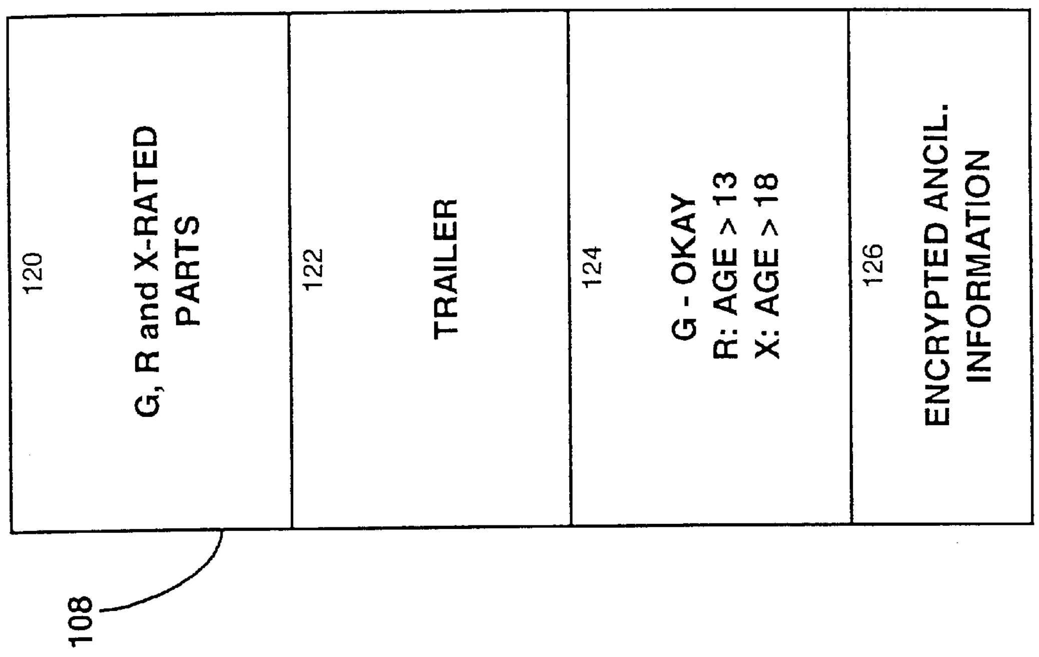


FIG. 19(b)

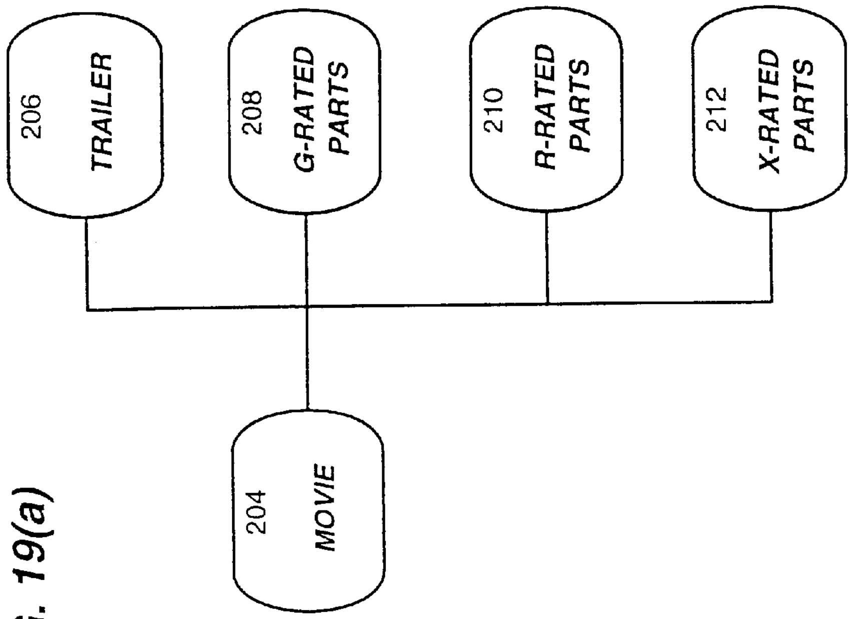


FIG. 19(a)

FIG. 19(d)

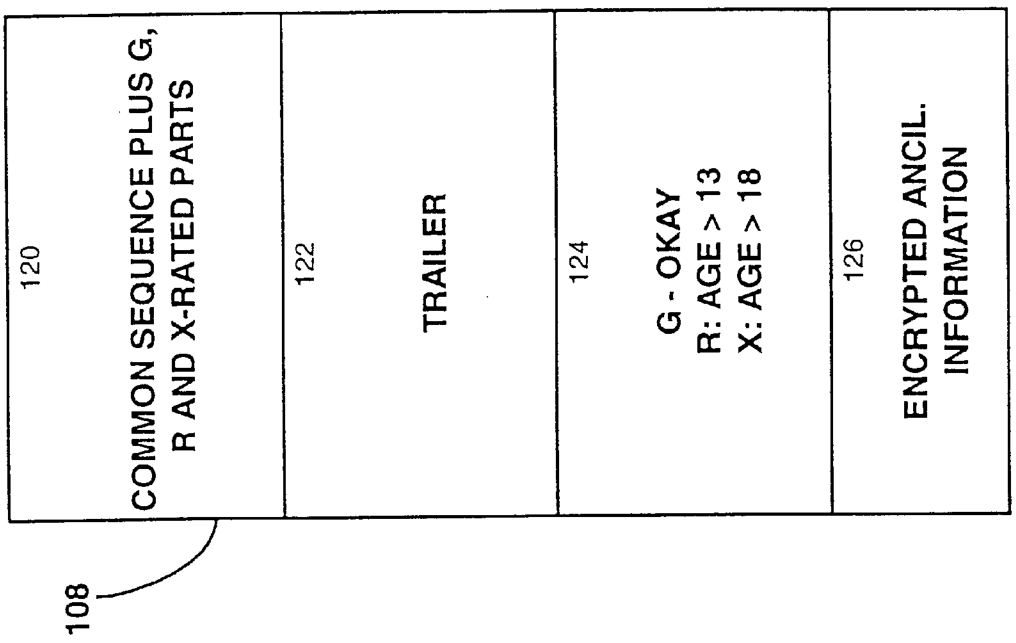
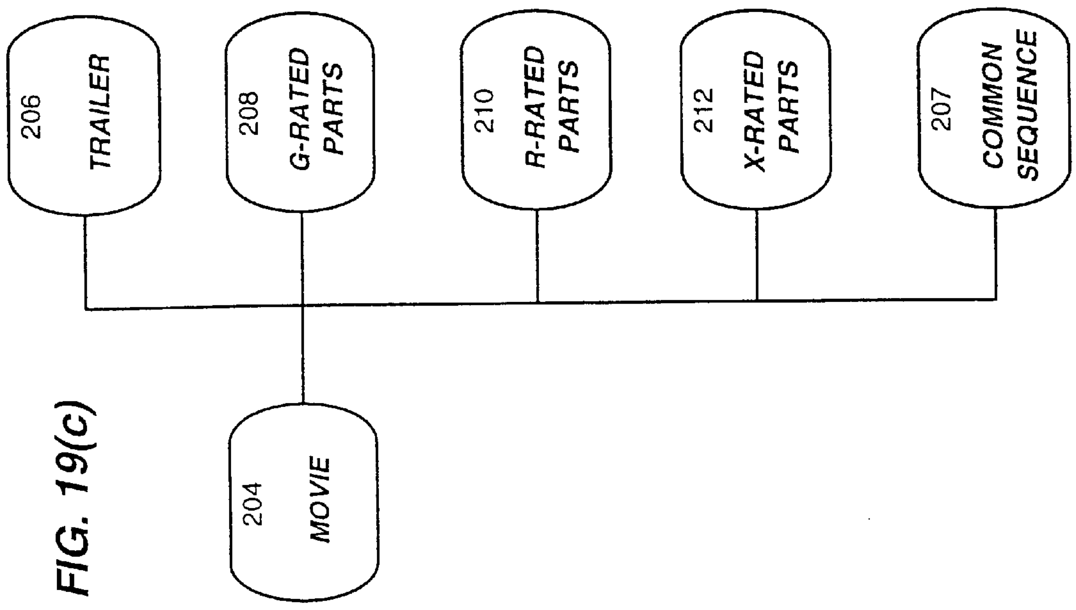

108
120
COMMON SEQUENCE PLUS G, R AND X-RATED PARTS
122
TRAILER
124
G - OKAY
R: AGE > 13
X: AGE > 18
126
ENCRYPTED ANCIL. INFORMATION

FIG. 19(c)



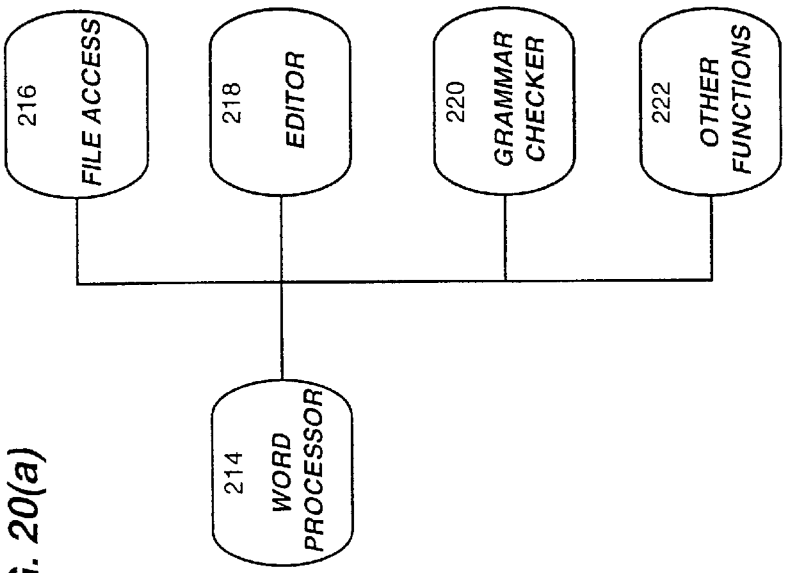


FIG. 20(a)

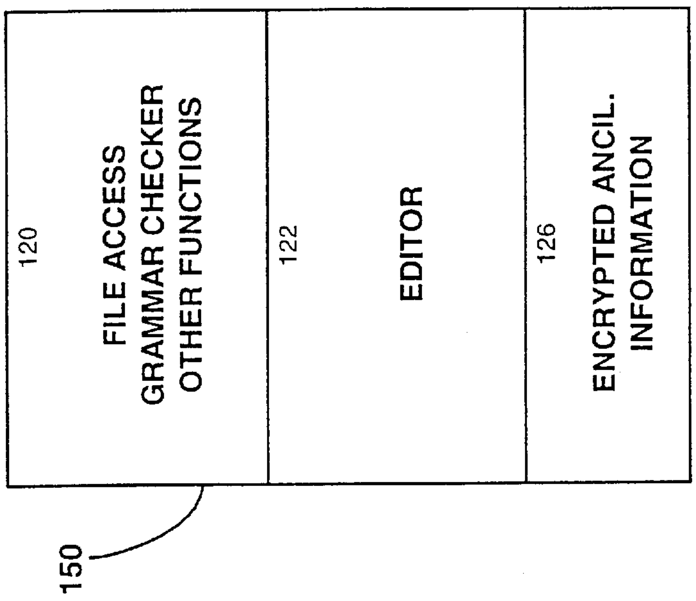

120 FILE ACCESS GRAMMAR CHECKER OTHER FUNCTIONS
122 EDITOR
126 ENCRYPTED ANCIL. INFORMATION
150

FIG. 20(b)

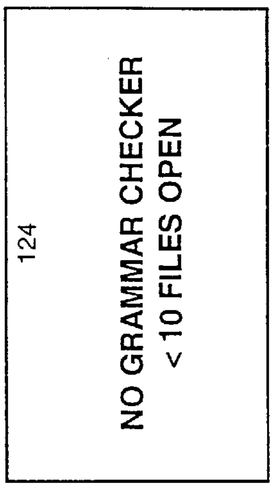


FIG. 20(c)

FIG. 21(a)

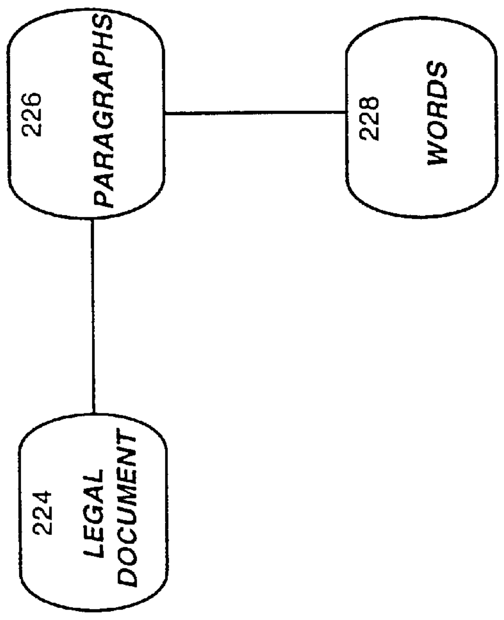


FIG. 21(b)

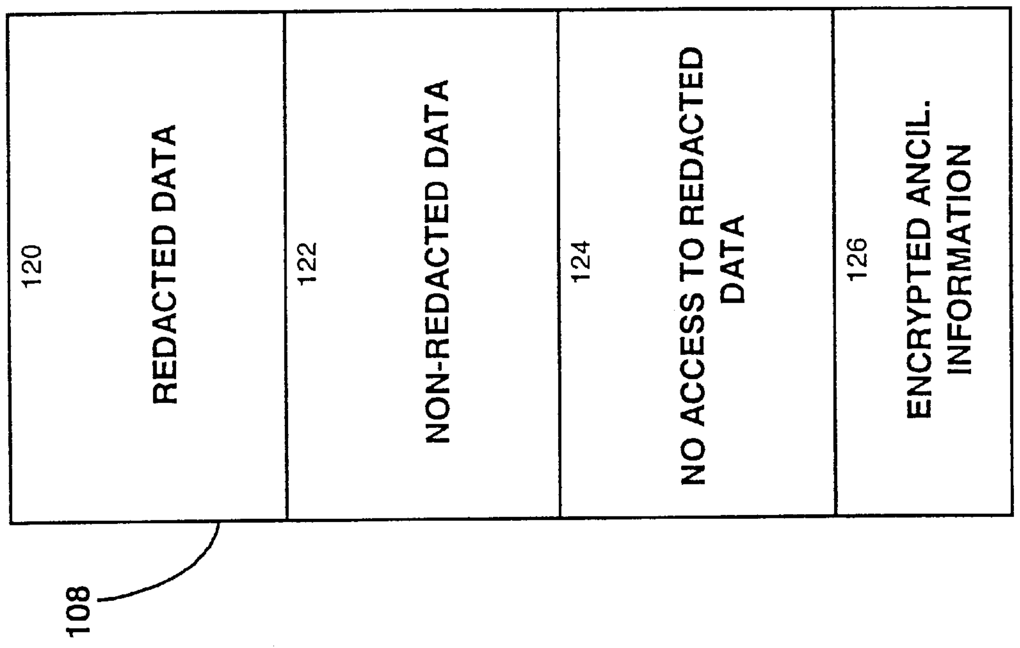


FIG. 22(b)

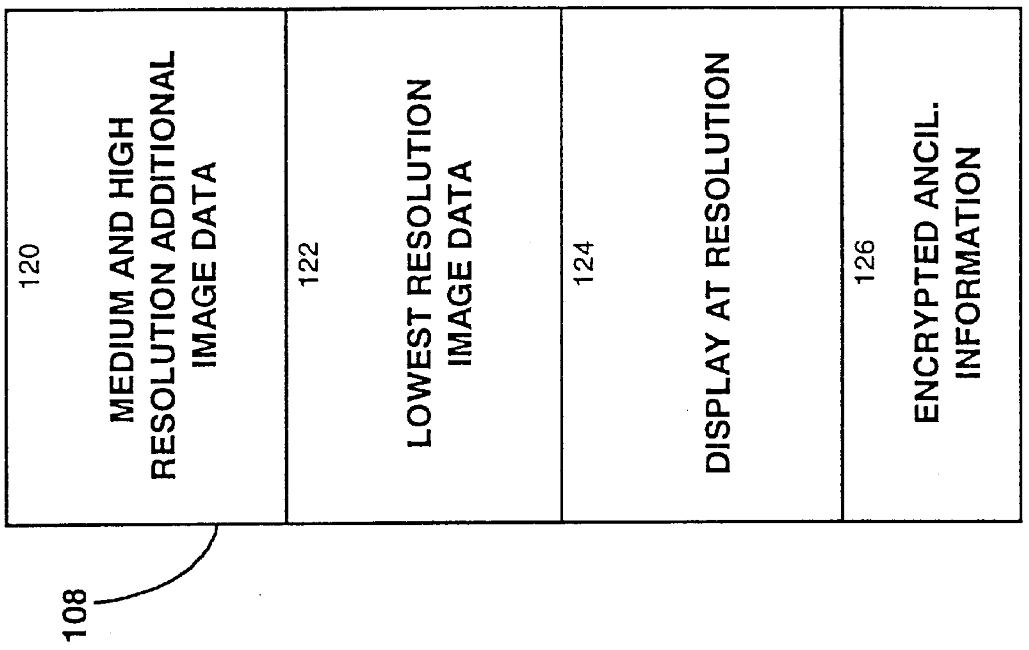
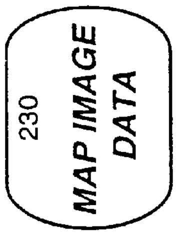

108
120
MEDIUM AND HIGH RESOLUTION ADDITIONAL IMAGE DATA
122
LOWEST RESOLUTION IMAGE DATA
124
DISPLAY AT RESOLUTION
126
ENCRYPTED ANCIL. INFORMATION

FIG. 22(a)



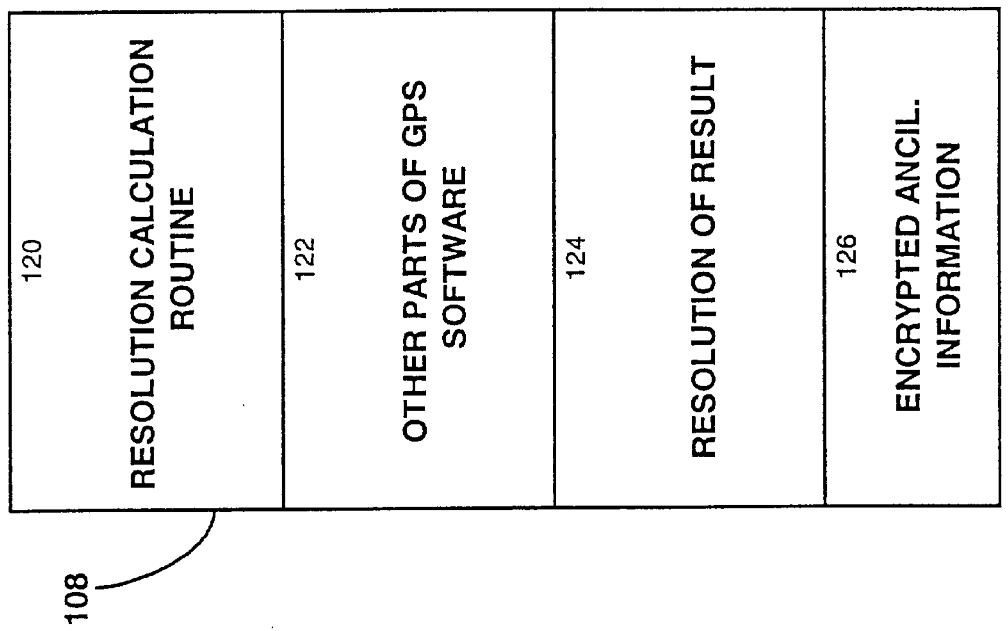

108
120
RESOLUTION CALCULATION ROUTINE
122
OTHER PARTS OF GPS SOFTWARE
124
RESOLUTION OF RESULT
126
ENCRYPTED ANCIL. INFORMATION

FIG. 23(b)

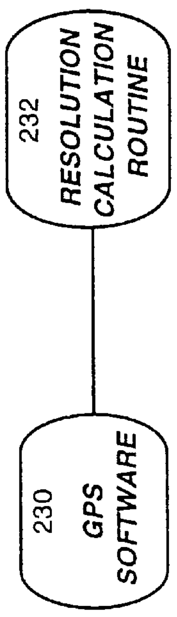


FIG. 23(a)

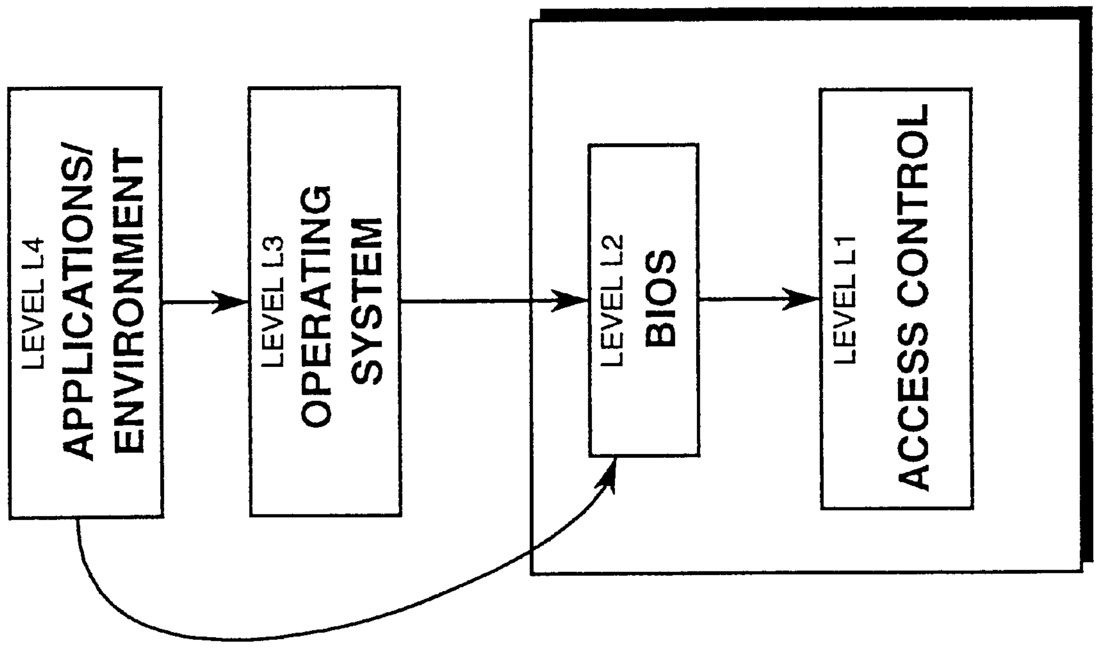

LEVEL L4
APPLICATIONS/ ENVIRONMENT
LEVEL L3
OPERATING SYSTEM
LEVEL L2
BIOS
LEVEL L1
ACCESS CONTROL

FIG. 24

SYSTEM FOR CONTROLLING ACCESS AND DISTRIBUTION OF DIGITAL PROPERTY

This is a division of application Ser. No. 08/968,887, filed Nov. 5, 1997 which is a continuation of Ser. No. 08/584,493, filed Jan. 11, 1996, now abandoned.

1. FIELD OF THE INVENTION

This invention relates to the control of distribution and access of digital property as well as to the payment therefor.

2. BACKGROUND OF THE INVENTION

The development and deployment of digital information networks is accompanied by new concerns for the protection of rights to data and information. The U.S. Congress Office of Technology Assessment identified the following key developments relevant to the area of this invention: there has been an overall movement to distributed computing; boundaries between types of information are blurring; the number and variety of service providers has increased. *Information Security and Privacy in Networked Environments*, Congress, Office of Technology Assessment, OTA-TCT-606, Washington, DC: U.S. Government Printing Office, September 1994.

Computer networks allow more interactivity; and, most significantly, electronic information has opened new questions about copyright, ownership, and responsibility for information. Technology, business practice, and law are changing at different rates, law arguably being the slowest.

Intellectual property, or information, is different from real property. A major difference between intellectual property and real property is that intellectual property can be embodied in forms which can be copied from the owner while the owner still retains the original. For example, a broadcast or performance of a musical composition can be recorded (and copies made of the recording) while the composer retains the original composition; a photograph can be reproduced while the owner retains the original negative.

In the past, when information was stored in analog form, the copying and redistribution of such information, while problematic, did not account for as much economic loss as is possible today. The storage of information in analog form uses a physical medium that is made to have some characteristic vary in proportion with the information to be stored. For instance, the groove on a vinyl record captures the frequency and intensity (volume) of a sound by the extent of its excursion. At each stage in the process of playing a record: the stylus tracing the groove, generation of a small voltage, amplification of the voltage, and reproduction of the sound, small errors are introduced. Today's high fidelity systems are very accurate, but they are not flawless.

Indeed, copying a vinyl record to a cassette tape results in a small, but noticeable, reduction in sound quality. If multiple generations of recording (e.g., cascaded recordings) were undertaken, the resulting product would be noticeably inferior to the original. Similarly, when multiple generations of photocopies of an image are made, the quality of the resulting image is typically poor, with many dark and light areas that were not present in the original image.

It is the inevitable gradual degradation of quality that has proven to be a practical disincentive to large scale copying of analog information. Notwithstanding this observation, where the potential profits are high, such copying is undertaken even though the resulting product's quality is significantly below that of the original. Videotape copies of movies

represent a good example. Some fraction of the marketplace is willing to accept a lower quality product in exchange for a significantly lower price. The logistics associated with making large numbers of copies (an inherently serial process), including obtaining the raw materials (cassettes), the reproduction equipment, and the distribution channels also have served to limit illicit production. Finally, the quality of the product as well as the markings on the package distinguish it from the original and may also serve as a disincentive (for some) to purchase an illicit copy.

Just as the invention of the printing press changed the way in which society interacted with information on paper, the technical advances in digital computers and communications in the closing years of the twentieth century have a potential for high impact on legal, moral, and business practice. The printing press is often credited as an enabling mechanism for the Renaissance and the Reformation in Europe. The advances in digital information technology will similarly impact commerce and law. Digital technology enables changing the representation of information without changing the content. (Of course the content can be changed too.)

The storage of information in digital form depends on the ability to encode information in binary form to arbitrary precision and to record that binary form in a physical medium that can take on two distinct characteristics. Preserving the fidelity of information recorded in binary (using media with two distinct and easily-differentiated characteristics) is easily accomplished. For instance, a compact disc stores information (each binary digit or bit) as the presence or absence of a hole (depression or pit) that reflects or does not reflect light. Compared to the analog recording of phonograph records, the information stored in each hole is unambiguously a binary digit, the value of which is either zero or one. No other values are possible. A digital tape stores each bit as a magnetic spot that is oriented either north/south or south/north. Today's digital sound systems use sufficiently many bits to capture sound levels beyond the ability of the human ear to distinguish a difference and in so doing attain so-called "perfect" fidelity.

A digital file can be copied with no loss of fidelity (as the mechanism need only distinguish between two easily-differentiated states). With straightforward and well-known error-correction mechanisms, even inevitable flaws can be made so improbable as to occur fewer than once in ten billion bits.

As a result of the ability to copy a file with no loss of fidelity, it is now almost impossible to differentiate a digital copy from the digital original. In a network environment recording materials, reproduction equipment and distribution are not impediments to copying. Consequently, in the digital domain the threshold inhibiting the making of illicit copies is significantly lowered. Evidence that this is the case is presented by the Software Publishers Association and by the Business Software Alliance, each of which indicates that billions of dollars of software is pirated (in the sense of being illicitly copied) each year. Additionally, print publishers hesitate to expand into the network marketplace because they are unable to control (in the sense of receiving compensation in return for rights) secondary distribution of their products as well as incorporation of their products into derivative products. Digitally stored information may include binary data, computer software, text, graphics, audio, and video. The uses of this information include news, entertainment, education, and analysis. Information may be distributed in many ways, including networks, magnetic media, CD-ROM, semiconductor memory modules, and wireless broadcast.

Copying and distributing large volumes of digital information over long distances is becoming easier and less costly. Such changes in cost and convenience of necessity impact business decisions concerning producing, distributing, promoting, and marketing. The commercial relationship among information producers (such as authors, performers, and artists), distributors (such as publishers, promoters, and broadcasters), and consumers must change in response to the technology.

The law concerning intellectual property is in ferment. Major revisions in the laws regarding the protection of computer programs have been suggested. *A Manifesto Concerning the Legal Protection of Computer Programs*, Samuelson, P. R. et al., *Columbia Law Review*, vol. 94, no. 8, pp. 2308–2431, December 1994. The European Union is working on harmonizing protection of intellectual property rights with respect to technology and differences in civil and common law countries. Commission of the European Union, Jul. 19, 1995, *Green Paper on Copyright and Neighboring Rights in the Information Society*, catalogue number CB-CO-95-421-EN-C, ISSN 0254-1475, ISBM 92-77-92580-9, Office for Official Publications of the European Communities, L-2985 Luxembourg. In the United States, the issue of protection of intellectual property rights is being addressed in the context of the National Information Infrastructure. The uncertainty of legal protection over time and from country to country only serves to emphasize the importance of and need for technical protection of intellectual property rights in information and data.

The principal technology which has been used for protecting intellectual property is cryptography. However, devising practical retail systems for delivery of intellectual property from distributor to consumer, as distinct from confidential transmission in national security and business activities among trusted and cleared personnel, has required innovation.

Executable software-based cryptography can ensure that data are distributed only to authorized users. The information to be protected is encrypted and transmitted to the authorized user(s). Separately, a decryption key is provided only to authorized users. The key is subsequently used to enable decryption of the information so that it is available to the authorized user(s).

Other ways of controlling access to portions of data or software have included the use of external devices or tokens (dongles) needed in order to access the data or selected features of a program. Possession of the token is made evident to the computer system by physical attachment of the token to the computer. A token is generally attached to a printer, game, or network port where executable software can check on its presence prior to authorizing access. Diskettes have also been used as dongles; their presence in the diskette drive is checked by the executing software. Because they must be actively interrogated, dongles are generally used to limit access to program features and not to limit access to information.

Of those prior art systems which make some use of encryption, none protects the data after it has been decrypted. Thus, secondary distribution and multiple uses are possible.

Further, in all of the prior art, access is all or nothing, that is, once access is granted, it cannot be controlled in any other ways. This makes it difficult to control copying, secondary distribution, as well as to obtain payment for all uses.

Originator controlled data dissemination is desirable. Several policies for control of dissemination of paper documents

are specified in *Control of Dissemination of Intelligence Information*, Directive No. 1/7, Director of Central Intelligence, 4 May 1981. This Originator-Controlled (ORCON) policy has motivated development of computerized access controls. ORCON requires the permission of the originator to distribute information beyond the original receivers designated by the originator. The Propagated Access Control (PAC) policy and the related Propagated Access Control List (PACL) were proposed as one way of implementing ORCON. "On the Need for a Third Form of Access Control," Graubart, R., *Proceedings of the 12th National Computer Security Conference*, pp. 296–303, 1989. Whenever an authorized subject reads an object with an associated PACL, that PACL becomes associated with the subject. Any new object created by the subject inherits the PACL. PACLs are associated with both subjects and objects.

Owner-Retained Access Control (ORAC) (described in "Beyond the Pale of MAC and DAC: Defining New Forms of Access Control," McCollum, C. J., et al. *Proceedings of the Symposium on Research in Security and Privacy*, IEEE Computer Society Press, 1990) is similar to PAC in propagating ACLs with non-discretionary enforcement. ORAC goes further, retaining the autonomy of all originators associated with a given object in making access decisions, while basing mediation of requests on the intersection of the access rights that have been granted. ORAC is motivated to implement several of the DCID 1/7 policies in addition to ORCON, namely NO_CONTRACTOR, NO_FOREIGN, and RELEASABLE_TO.

Originator-Controlled Access Control (ORGCON) (described in "Generalized Framework for Access Control: Towards Prototyping the ORGCON Policy," Abrams, M. D., et al. *Proceedings of the 14th National Computer Security Conference*, October 1991) is a strong form of identity-based access control—it explicitly defines authority and delegation of authority, provides for accountability, and has an explicit inheritance policy. In ORGCON, the distribution list is indelibly attached to the object (i.e., the distribution list cannot be disassociated from the object, even in the limited cases where copying is permitted). ORGCON is a read, no-copy policy. Its formal model (taught in "A Rule-Set Approach to Formal Modeling of a Trusted Computer System," LaPadula, L. J., *Computing Systems Journal*, Vol. 7, No. 1, pp. 113–167, Winter 1994) distinguishes among device types in order to deal with the policy that no storage copy of an object is permitted. Information may be copied only to the display and printer, but not to any other device types.

The Typed Access Matrix (TAM) Model (described in "The Typed Access Matrix Model," Sandhu, R. S., *Proceedings of the Symposium on Research in Security and Privacy*, IEEE Computer society, pp. 122–136, 1992; and "Implementation Considerations for the Typed Access Matrix Model in a Distributed Environment," Sandhu, R. S., and G. S. Suri, 1992, *Proceedings of the 15th National Computer Security Conference*, pp. 221–235) incorporates strong typing into the access matrix model to provide a flexible model that can express a rich variety of security policies while addressing propagation of access rights and the safety problem. The safety problem is closely related to the fundamental flaw in Discretionary Access Control (DAC) that malicious code can modify the protection state. Types and rights are specified as part of the system definition; they are not predetermined in TAM.

The prior art, including cryptographic processes, tokens, dongles, so-balled "uncopyable" media, various executable software protection schemes, and executable software for printing that places an identifier on all printed output in a fashion not apparent to a human, fails to limit either secondary distribution or distribution of derivative works.

This shortcoming is not a failure of mechanism, but rather it is an architectural design omission. The problem of copying by the authorized user is simply not addressed. In each case, once the data are available to an authorized user, they are basically unprotected and may be copied, modified, or transmitted at will. Schemes that include identifiers on printed material, although they may aid in identifying the source of copied material, do not prevent secondary distribution.

Executable software-based cryptography can ensure that data are distributed only to authorized users. However, once data are received they may be freely manipulated and redistributed.

The information to be protected is encrypted and transmitted to the authorized user(s). In some systems the encrypted information is made freely available. Separately, a decryption key is provided only to authorized users. The key is subsequently used to enable decryption of the information so that it is available to the authorized user(s). It is at this point that the information is subject to manipulation and redistribution without further limitation.

As mentioned above, a dongle or token can be used to authorize access to executable software. However, once access has been granted to information that information is subject to manipulation and redistribution without further limitation. Further, dongles have proven to be unpopular because of the need to keep track of them and ensure that they are separately secured.

Uncopyable media, generally used either to control distribution of information or to control usage of executable software, are unpopular because of the user's inability to create a backup copy. Further, most so-called uncopyable disks have fallen victim to general-purpose duplication programs, rendering their protection useless. Sometimes, as in early releases of Lotus 1-2-3, an uncopyable disk was provided with the executable software release and had to be inserted in a floppy-disk drive for the executable software to function (operating as a disk dongle). Users soon learned how to by-pass the executable software so that the disk need not be present. Even where partially effective, the uncopyable disk did not serve as a deterrent to capturing information and redistributing it.

The degree of protection of data is typically made by the data owners and/or distributors based on their security analysis. It is common to perform security analysis in terms of risks, threats, vulnerabilities, and countermeasures. An owner's estimate of the probability that a particular threat will materialize is crucial to selecting appropriate rules to protect property rights.

Threat can be characterized as the intensity of attack on the data, which can be described as low, medium, and high.

Low	For a security function to be rated as "suitable for use in a low threat environment," it shall be shown that the security function provides protection against unintended or casual breach of security by attackers possessing a low level of expertise, opportunities, resources and motivation. However, such a security function may be capable of being defeated by a knowledgeable attacker.
Medium	For a security function to be rated as "suitable for use in a medium threat environment," it shall be shown that the security function provides protection against attackers possessing a moderate level of expertise, opportunities, resources and motivation.
High	For a security function to be rated as "suitable for use in a high threat environment," it shall be shown that the security function provides protection against attackers possessing a high level of expertise, opportunity, resources and motivation. A successful attack is judged as being beyond normal practicality.

The following list covers some common anticipated threats to data and processing systems.

Threat: Capture of Output Signal

No matter what method is used to protect a data file, the data stored therein can be captured as a signal en route to an output device. Capture of an analog output results in some degradation of signal quality. But the market for bootleg copies of videos, for example, appears to be insensitive to such quality if the price is right. A captured digital signal suffers degradation of quality only as a result of bit errors (i.e., if the data capture was not completely accurate).

This threat is well known to the entertainment industry. Various approaches to protection have been incorporated in set-top boxes discussed in "Inside the Set-Top Box," Ciciora, W. S., *IEEE Spectrum*, pp. 70–75, April 1995.

Threat: Digital Copying

Once data have been decrypted, the resulting cleartext must be protected from unauthorized copying. Creating an unauthorized local copy, or disseminating the data without authorization each results in an original-quality copy without compensation to the owner.

Threat: Deliberate Attack Via Legacy (Pre-Existing) and Customized Hardware

High-intensity attack by attackers possessing a high level of expertise, opportunity, resources and motivation must be considered. Attackers in this category might include foreign governments and industrial espionage agents, teenage crackers, and resellers of pirated intellectual property. One manifestation of this threat is in uncontrolled hardware. The nominally protected information would be available in the memory and could be accessed via dual-ported memory or even by DMA (direct memory access) from a peripheral.

A strong indication of the usefulness and desirability of the present invention can be found in the legislation pending before the U.S. Congress to make illegal the by-passing or avoiding of copyright protection schemes. See S.1284, 104th Congress, 1st sess. (1995).

It is desirable to have a system of distributing data (intellectual property) that prevents copying, restricts re-distribution of the data and provides controlled access to the data.

SUMMARY OF THE INVENTION

This invention controls access to and use and distribution of data.

For example, when the data are in the form of textual and graphical information, this invention can control how much of the information is displayed and in what form; or, when the data represents a computer software program, this invention can control how much of the software's functionality is available. Classified data are similarly controlled.

In addition, this invention controls secondary distribution and creation of derivative works. Prior art systems rely on software for security. Without the tamper detection/reset mechanism of this invention, software can be modified or data can be intercepted rendering useless any attempts at control.

Degrees of protection utilized in the computer system hardware (for example, tamperproof and tamper-detect features) and the cryptographic tools will depend on the nature of the data to be protected as well as the user environment.

In one preferred embodiment, this invention is a method of controlling access to data by protecting portions of the data; determining rules concerning access rights to the data; preventing access to the protected portions of the data other than in a non-useable form; and permitting a user access to the data only in accordance with the rules as enforced by a tamper detecting mechanism.

In another preferred embodiment, this invention is a device for controlling access to digital data, the digital data comprising protected data portions and rules concerning access rights to the digital data. The device includes storage means for storing the rules; and means for accessing the protected data portions only in accordance with the rules, whereby user access to the protected data portions is permitted only if the rules indicate that the user is allowed to access the portions of the data.

In another aspect, this invention is a method of distributing digital data for subsequent controlled use of the data by a user. The method includes protecting portions of the digital data; preventing access to the protected portions of the data other than in a non-useable form; determining rules concerning access rights to the data; protecting the rules; and providing the protected portions of the digital data and the protected rules. The user is provided controlled access to the data only in accordance with the rules as enforced by a tamper detecting access mechanism.

In another aspect, this invention is a storage device, readable by a machine, tangibly embodying a package of digital data comprising protected portions of digital data; and rules concerning access rights to the digital data, whereby a user is provided controlled access to the digital data only in accordance with the rules as enforced by a tamper detecting access mechanism.

The data represent computer software, text, graphics, audio, and video, alone or in combinations.

The protecting is done by encrypting the portions of the data, and access is prevented to the encrypted portions of the data other than in encrypted form.

In some embodiments the rules are provided with the data, whereas in others the rules are provided separately. The rules can specify various access rights and controls, including rights of further distribution of the data.

In preferred embodiments, data are destroyed when tampering is detected.

The device containing the mechanism of the present invention can be a stand-alone device such as a facsimile machine, a television, a VCR, a laser printer, a telephone, a laser disk player, a computer system or the like.

As noted above, the rules, policies and protections of data are typically made by the data owners and/or distributors based on their security analysis of various threats. The various threats listed above are dealt with by countermeasures in the present invention.

Threat: Capture of Output Signal

Countermeasure: Encrypt or Scramble Output Signal

Protection of the output signal is accomplished with encryption of a digital signal (as is done in the present invention) and scrambling of an analog signal. This solution requires installing decryption or unscrambling capability in the output device, TV or monitor, along with appropriate tamper-detection capability. Encryption or scrambling might be effected using a public key associated with the output device (although, to prevent so-called "spoofing," obtained from a certification authority and not from the output device). Alternatively, the output might be encrypted or scrambled using a private key only available to the designated output device (again ensured via some certification mechanism). The output signal is decrypted or unscrambled by the output device using its private key and is not available in plaintext form outside of the device's protected enclosure.

Countermeasure: Protect Output Signal by Packaging

The output signal is protected by making it unavailable outside the access mechanism. A sealed-unit computer with tamper detection provides the necessary protection. Examples of the acceptability of such packaging include lap-top computers and the original Macintosh computer, as well as integrated televisions, VCRs and video or audio laser disk players.

Threat: Digital Copying

Countermeasure: Secure Coprocessor

Selection of a secure coprocessor is indicated to implement protection against unauthorized use when an operating system (OS) is determined to be untrustworthy—that is, when the OS cannot provide adequate resistance to the anticipated threat. When the OS is untrustworthy, any measures implemented in the OS, or protected by it, can be circumvented through the OS or by-passing it.

Countermeasure: Detection of Unsealing

The protection provided by a coprocessor could be circumvented by tampering. The coprocessor is protected by tamper detection that causes the rules, cryptographic data, and decrypted protected data to be destroyed. Both passive and active means are used to effect such destruction. Semiconductor memory is volatile and does not retain data when power is removed. A long-life battery provides energy sufficient to allow rewriting (zeroizing) nonvolatile memory containing, for example, the private key. Without the private key the system will be unable to decrypt any protected data and it must be returned to an authorized service facility for installation of a new private key.

Threat: Deliberate Attack Via Legacy and Customized Hardware

Countermeasure: Keep the Information on the Coprocessor Board

Access may be controlled if the information leaves the coprocessor board only for output purposes. Deciphered information is retained in memory on the coprocessor board, not in main memory. Program execution occurs in the coprocessor on the board (e.g, operating in the same manner as did so-called "accelerator" coprocessors that allowed a user to install an 80286 processor in an 80186 system, allowing the user to shift all functions to or from the faster coprocessor using a software command). Where information must leave the coprocessor board, e.g., to be sent to an output device, it may, depending on the associated rules, be encrypted. To receive and process encrypted data, the output device must have an access mechanism as well as public and private keys and tamper detect capability. Because some output peripheral devices do not have the capability of retransmission, the device may be a subset of the full access mechanism associated with a processor or computer system.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and other objects and advantages of the invention will be apparent upon consideration of the following detailed description, taken in conjunction with the accompanying drawings, in which the reference characters refer to like parts throughout and in which:

FIG. **1** is a schematic block diagram of an embodiment of a digital data access and distribution system according to the present invention;

FIGS. **2** and **3** show logical data structures used by the system depicted in FIG. **1**;

FIG. **4** is a flow chart of the authoring mechanism of the embodiment of the present invention depicted in FIG. **1**;

FIG. **5** is a schematic block diagram of another embodiment of a digital data access and distribution system according to the present invention;

FIG. **6** is a logical data structure used by the embodiment depicted in FIG. **5**;

FIG. **7** is a flow chart of the authoring mechanism of the embodiment of the present invention depicted in FIG. **5**;

FIGS. **8** and **9** show schematic block diagrams of embodiments of the access mechanism according to the present invention;

FIGS. **10**(*a*)–**13** are flow charts of the data access using the access mechanisms shown in FIGS. **8**, **9** and **15**;

FIG. **14** shows an embodiment of the invention which uses an external user status determination mechanism;

FIG. **15** is a schematic block diagram of an embodiment of a distribution system for derivative works according to the present invention;

FIG. **16** is a flow chart of data access using the access mechanism shown in FIG. **15**;

FIGS. **17**(*a*) and **17**(*b*) show packetized data according to the logical data structures shown in FIGS. **2** and **6**;

FIGS. **18**(*a*)–**23**(*b*) show various examples of data and their packaging according to the present invention; and

FIG. **24** shows various implementation levels of a typical computer system employing an access mechanism according to the present invention.

DETAILED DESCRIPTION OF THE PRESENTLY PREFERRED EXEMPLARY EMBODIMENTS

A schematic block diagram of a presently preferred exemplary embodiment of a digital data access and distribution system **100** according to the present invention is depicted in FIG. **1**. System **100** includes two main components: a data distributor **102** and a user **104**. The data distributor **102** takes data **106** and produces packaged data **108** which are provided to the user **104** via communication channel **105**, perhaps in return for some form of payment **110**.

Corresponding to each of the distributor **102** and the user **104** are the system's authoring mechanism **112** and access mechanism **114**, respectively. The authoring mechanism **112** of the distributor **102** takes the data **106** to be packaged and produces packaged data **108** which is provided to user **104** by a distribution mechanism **118**. The packaged data **108** may include access rules **116** in encrypted form encoded therewith, or the access rules **116** may be provided to the user **104** separately (as shown in the embodiment of FIG. **5**).

The access mechanism **114** of the user **104** takes the packaged data **108**, either including an encrypted version of the access rules **116** or having the access rules provided separately, and enables the user to access the data in various controlled ways, depending on the access rules.

Data **106** provided to or generated by the distributor **102** can be any combination of binary data representing, for example,.computer software, text, graphics, audio, video and the like, alone or in combinations. As described below (with respect to the embodiment shown in FIG. **15**), in some embodiments data **106** can also include other packaged data produced by an authoring mechanism according to this invention.

The difference between the embodiments of the distributors **102** and **190**, shown in FIGS. **1** and **15**, respectively, is that the distributor **102** (FIG. **1**) does not include an access mechanism **114**. Accordingly, distributor **102** deals only with newly created data (that is, with non-derivative data). The embodiment shown in FIG. **15** (discussed below) includes the functionality of the embodiment shown in FIG. **1**, and can also deal with input of protected data (previously packaged by a distributor). The embodiment of distributor **102** shown in FIG. **1** can be implemented purely in software (depending on the trust level of the employees of the publisher), whereas the embodiment of distributor **190** shown in FIG. **15** requires some hardware implementation.

Data **106** can also be provided to the distributor in non-digital form and converted to digital form by the distributor in a known and suitable fashion. The content of the data **106** can include, for example, news, entertainment, education, analysis and the like, alone or in combinations.

Note, as used herein, computer software refers to any software program used to control any computer processor. This includes, but is in no way limited to, processors in stand-alone computers; processors in video and audio devices such as televisions, video recorders and the like; processors in output devices such as printers, displays, facsimile machines and the like; and processors in appliances, automobiles, telephones and the like.

The data **106** are typically intellectual property subject to control. In some cases, distributor **102** may receive some form of payment **110** from the user **104** for accessing the data. This payment, or some part thereof, may then be provided directly to the actual owner (not shown) of the data **106**. Further, the payment or part thereof may be made before, during or after use of the data.

As noted above, the packaged data **108** may include an encrypted version of the access rules **116**, or these rules may be provided to the user separately. The logical data structure for the packaged data **108** is shown in FIG. **2** and includes an encrypted body part **120**, an unencrypted body part **122**, encrypted rules **124** (if provided with the packaged data), and encrypted ancillary information **126**. Encrypted rules **124** are an encrypted version of access rules **116**.

The actual format and layout of the data is dependent on the type of data, their intended use, the manner in which they are to be accessed and the granularity of control to be exercised on the data. An encyclopedia, for example, would likely be organized differently from a movie or a musical selection. Since the data can be any combination of binary data, different parts of the packaged data **108** may be structured differently, as appropriate. Accordingly, encrypted body part **120** is potentially made up of encrypted body elements, and similarly, unencrypted body part **122** is potentially made up of unencrypted body elements.

It is, however, envisaged that in presently preferred embodiments the data will be structured such that some data parts or elements have header information which enables the data to be traversed or navigated according to whatever rules are to be applied and in a manner appropriate for those data.

An example of the structure of rules **116** is shown in FIG. **3**, wherein the rules include various forms of validity checking and identification information such as version number **127**, authentication data **128**, license number **130**, intellectual property identifier **132**, first and last valid generations of the product **134**, **136**. The rules **116** further include an encrypted data key **138** as well as the actual rules **140**, **142**, **144–146** to be applied when access is made to the data by a user. The actual rules include, but are not limited to, standard, extended and custom permissions **140**, **142**, **144–146**, and co-requisite rules (permission lists) of source data **145**.

The function of each field in the rules shown in FIG. **3** is given in TABLE I, below.

TABLE I

Field	Function
Version number 127	Defines internal configuration template
Authentication (hash) 128	Validates integrity of this data file.
License number of these rules 130.	Used by publisher to identify owner.
Intellectual property identifier 132.	Identifies the intellectual property product.
First valid generation of the product 134.	Defines extent of validity of the license.
Last valid generation of the product 136.	Defines extent of validity of the license.
Encrypted data key 138.	Key to access the data.
Standard permissions 140.	List of basic access permissions for data.
Extended permissions 142.	List of extended access permissions for data.
Custom permissions 144.	Executable code modules.
Co-requisite rules (permissions) for source data 145.	Indicates which source data rules are needed.
Token/biometrics 146	Indicates the physical tokens and/or biometric characteristics (if any) required for identification of each authorized user.
System IDs/Public keys 147	Other systems to which these rules may be redistributed.

A complete introduction and references to further reading concerning cryptography and cryptographic techniques and mechanisms are found in Abrams, M. D. and Podell, H. J., "Cryptography," *Security-An Integrated Collection of Essays*, Abrams, M. D. et al, eds. IEEE Computer Society Press, 1995, which is hereby incorporated herein by reference.

The Authoring Mechanism

As shown in FIG. **1**, the authoring mechanism **112** of the distributor **102** takes data **106** and produces packaged data **108** for distribution. The process of producing the packaged data which includes rules **116** is described with reference to FIGS. **1–4**.

The authoring mechanism **112** incorporates existing source data **106** into a packaged format for dissemination. As noted above, data **106** can include but are not limited to combinations of computer software, text, graphics, audio, video and the like. The data **106** may be provided to the authoring mechanism **112** in various proprietary data formats used in vendor software packages as well as having lower level formats for graphics, tables, charts, spreadsheets, text, still and motion pictures, audio and the like.

Using the authoring mechanism **112**, those elements of the data **106** that are to be encrypted are selected, as are the cryptographic algorithms and protocols to be employed, the payment procedures for the use of the data, and other decisions governing how the user **104** will be permitted to use the data. These decisions are used in constructing the permission lists to be included in the rules **116**. Different classes of users can be defined, based, for example, on age, fee paid, qualifications and the like.

The presently preferred embodiment employs asymmetric encryption algorithms in the authoring and access mechanisms. The keys for these algorithms are protected within the system and are never exposed. The data-encrypting key, K_D, is the same for all copies of the data. K_D is selected by the distributor **102** and may be different for each product (i.e., for each packaged data **108**). The symmetric encryption algorithm used for encrypting the data is associated with K_D and may also be selected by the distributor. K_D is encrypted using a rule-encrypting key K_R. When the rules are distributed with the product (packaged data **108**), K_R is the same for all products and all embodiments of the system. When the rules are distributed separately from the product, K_R can be unique for each version of the system. The rule-encrypting key K_R is known only to (and protected within) each receiving computer of each user.

With reference to FIG. **4** which shows a flow chart of a version of the authoring mechanism of the present invention in which the rules are distributed with the packaged data **108**, the distributor **102** (acting as a representative of the owner of the data **106**) selects a data-encrypting algorithm (DEA) (step **S400**) and data-encrypting key K_D (step **S402**), and encrypts the data-encrypting key K_D using K_R (step **S404**). The encrypted data-encrypting key K_D is then stored in the encrypted ancillary information **126** of the packaged data **108** (in step **S406**).

The algorithm selection (in step **S400**) is based on an assessment of risk, the degree of protection desired as well as other factors such as speed, reliability, exportability and the like. As used herein, risk refers to the expected loss due to, or impact of, anticipated threats in light of system vulnerabilities and strength or determination of relevant threat agents. Alternatively, risk can refer to the probability that a particular threat will exploit a particular vulnerability of the system. An analysis of risk, threats and vulnerability is provided below. Examples of possible data-encryption algorithms include, but are not limited to, DES, RSA, PGP and SKIPJACK. The system may use a preferred encryption algorithm and may also provide a mechanism for using algorithms provided with the data **106** by the owner of the data.

The data-encrypting key K_D may be generated in a typical manner, suitable for the selected data-encrypting algorithm. For data having lower value to its owner, or having lower risk of loss, all distributions may rely on a single data-encrypting key (or perhaps a small number of data-encrypting keys). Another encryption method, uses a unique data-encrypting key for each item of data to be distributed.

Having selected a data-encrypting algorithm and key, K_D, (**S400–S402**) and having encrypted and stored the key (**S404–S406**), the distributor **102** proceeds to process the various elements of the data **106**. The data are processed at a granularity dependent on the type of restrictions needed on their use and on the form of the data themselves, that is, the form in which the data have been provided. The distributor obtains (step **S407**) and examines each part or element of the data (at the desired granularity) and determines whether or not the element being processed (the current element being examined) is in the body of the data (step **S408**) (as opposed to being rules or ancillary information). If the current element being examined is determined to be in the body of the data, the distributor then decides whether or not the current data element is to be protected (step **S410**), that is, whether or not access to that element of the data is to be controlled and the data element is to be encrypted.

If the current data element is not to be protected, it is stored (step **S412**) in the unencrypted body part **122** of the packaged data **108**. Otherwise, if the current data element is to be protected, it is encrypted using the data-encrypting key K_D (step **S414**) and then the encrypted current data element is stored in the encrypted body part **120** of the packaged data **108** (step **S416**), after which the next element is processed (starting at step **S407**).

For example, if the data **106** are a textual article, the abstract of the article might not be protected (encrypted) while the rest of the article would be.

If the current data element is determined not to be in the body of the data (step **S408**), the distributor then determines if the current data element is access rules provided by the data owner (step **S418**). If so, the rules are protected by encrypting them using the rule-encrypting key K_R (step **S420**) and the encrypted rules are then stored in the encrypted rules part **124** of the packaged data **108** (step **S422**).

If the current data element (being processed) is not access rules, the distributor determines whether or not it is ancillary information (step **S424**). This information includes such things as the identification of the publisher and the like. If the current data element is determined to be ancillary information, the ancillary information is protected by encrypting it using the data-encrypting key K_D (step **S426**) and then the encrypted ancillary information is stored in the encrypted ancillary information part **126** of the packaged data **108** (step **S428**).

If the data are rules or ancillary information to be encrypted, then, after appropriate processing, the next data element is processed (step **S407**).

If the current data element is not a body part, access rules or ancillary information, some form of error is assumed to have occurred and is processed (step **S430**). After the error has been processed, the mechanism can continue processing the next data element (step **S407**) or terminate, depending on the implementation.

The operation of the system **101** shown in FIG. **5** differs from system **100** of FIG. **1** in that the rules **116** are distributed to users **104** separately from the packaged data **108**. This is achieved with an authoring mechanism **148** which takes as input data **106** and rules **116** and produces, separately, packaged data **150** and packaged rules **152**. The packaged data **150** without the rules has the form shown in FIG. **6**, which is essentially the same as the structure shown in FIG. **2**, but without the encrypted rules **124**.

Note that an hybrid system, wherein some rules are packaged with the data and other rules are packaged separately is foreseen, using a combination of the mechanisms shown in FIGS. **1** and **5**. In such a system, an operator selects which mode of operation to employ.

FIG. **7** shows a flow chart of a version of the authoring mechanism **148** of the present invention in which the rules **116** are distributed by distributor **102** separately from the packaged data **150**. Rules **116** and data **106** can be presented to the authoring mechanism **148** in any order, or in an

interleaved fashion. In fact, the rules **116** need not all be provided together. The distributor **102** first selects a data-encrypting algorithm and a data encrypting key, K_D (step **S700**). Then the authoring mechanism **148** processes the data element-by-element (starting at step **S702**). As in the case of the mechanism shown in FIG. **4**, a data element is assumed to be one of either a body part, ancillary information or access rules.

First it is determined whether or not the current data element is a body part (step **S716**). If it is determined (in step **S716**) that the current data element is a body element, then it must be determined (in step **S718**) whether or not the data are to be protected. As in the case when the rules are distributed with the packaged data **108**, the decision as to whether or not to protect a specific data element depends on the owner of the data and the distribution policies as implemented in the rules.

If the data are to be protected (step **S718**), the data in the current data element are encrypted using data-encrypting key K_D (step **S720**) and then the encrypted data are stored in the packaged data **150** in the encrypted body part section **120** (step **S722**). On the other hand, if the data in the current data element are not to be protected, the data are stored in the unencrypted body part section **122** of the packaged data **150** (in step **S724**). In either case, after the data element is stored (steps **S722** or **S724**), the next data element is processed (starting at step **S702**).

If the current data element is determined not to be a body element (step **S716**), then the mechanism checks to determine whether or not the current data element is ancillary information (step **S726**). If the current data element is determined to be ancillary information, it is protected by encrypting it using data-encrypting key K_D (step **S726**) and then the encrypted current data element is stored in the packaged data **150** in the encrypted ancillary information section **126** (in step **S730**). Then the next data element is processed, starting at step **S702**.

If the current data element is neither a body element (step **S716**) nor ancillary information (step **S726**), then the it is determined whether or not the current data element is access rules (step **S732**). If so, the rules are to be distributed separately from the packaged data **150**, and are processed accordingly as follows:

If this is the first time the access mechanism is processing rules for this data set then a rule-encrypting key K_R must be determined. Accordingly, it is determined whether these are the first rules being processed for this data set (step **S734**). If so, obtain and validate the serial number, SN, of the system (steps **S736** and **S738**). Then calculate the rule-encrypting key K_R as a function of the validated serial number (K_R=f(SN), for some appropriate function f (step **S740**). Function f may, for example, be an inquiry to a certification database or certification authority to obtain the public key so as to ensure that the serial number is authentic. Having determined the rule-encrypting key (step **S740**), encrypt the data key K_D with the calculated rule-encrypting key K_R (step **S742**) and store the keys (step **S744**). Next, encrypt the rules using the rule-encrypting key K_R (step **S746**). The encrypted rules and the encrypted data key K_D are stored as packaged rules **152** for subsequent distribution. The rule-encrypting key K_R may be stored or recalculated from the serial number whenever needed.

If it is determined (in step **S734**) that the this is not the first rules being processed for this data set, then the rule-encrypting key K_R has already been calculated (step **S740**) and stored (step **S744**). In that case, the rules in the current data element are encrypted using the rule-encrypting key K_R (step **S742**).

Once the rules in the current data element are processed, processing continues with the next data element (step **S702**).

If the authoring mechanism **148** determines that the current data element is not a body part (step **S716**), ancillary information (step **S726**) or rules (step **S732**), then some form of error has occurred and is processed (step **S748**). After an error has occurred, the mechanism **148** can either cease processing (step **S750**) or, in some embodiments, continue processing further data elements (step **S702**).

The data **106** provided to the distributor **102** and the packaged data **108** (or **150** and packaged rules **152**, if provided separately) provided to the user **104**, may be provided and distributed in various ways, including but not limited to, via digital communications networks (for example, the Internet or the projected National Information Infrastructure (NII)), magnetic media (for example, tape or disk), CD-ROM, semiconductor memory modules (for example, flash memory, PCMCIA RAM cards), and wireless (for example, broadcast). The packaged data **108** may be provided to a user as a single packaged entity or as a continuous stream of data. For example, a user may obtain a CD-ROM having a movie stored as packaged data thereon or the user may obtain the movie as a continuous stream of broadcast data for one-time viewing.

Information (such as the packaged data **108** from the distributor **102** to the user **104**) can be transmitted openly, that is, using mechanisms and media that are subject to access and copying. In other words, communication channel **105** may be insecure.

The Access Mechanism

The access mechanism **114** allows a user **104** to access the data in packaged data **108** (or **150**) according to the rules provided with (or separately from, as packaged rules **152**) the packaged data and prevents the user or anyone else from accessing the data other than as allowed by the rules. However, having granted a user controlled access to data (according to the rules), it is necessary to prevent the user or others from gaining unauthorized access to the data. It is further necessary to prevent the data from being further distributed without authorization.

The access mechanism **114** used by the user **104** to access data is described with reference to FIG. **8** and includes a processing unit **154**, read-only memory (ROM) **156**, volatile memory (RAM) **158**, I/O controller **165** and some form of energy source **166** such as, for example, a battery. Access mechanism **114** may also include electrically-alterable non-volatile memory **160**, a hard disk **162**, a display **164**, and special purpose components such as encryption hardware **168**.

The access mechanism **114** is also connected via insecure channels **174** and **176** and I/O controller **165** to various controlled display or output devices such as controlled printer **178** and controlled display monitor **180**. (Interaction with these controlled devices is described in detail below.)

Various other devices or mechanisms can be connected to I/O controller **165**, for example, display **155**, printer **157**, network connection device **159**, floppy disk **161** and modem **163**. These devices will only receive plaintext from the I/O controller **165**, and then only such as is allowed by the rules. The network connection device **159** can receive either plaintext or encrypted text for further distribution.

All components of the access mechanism **114** are packaged in such a way as to exclude any unknown access by a user and to discover any such attempt at user access to the components or their contents. That is, the access mechanism **114** is packaged in a tamper-detectable manner, and, once tampering is detected, the. access mechanism is disabled.

The line **167** depicted in FIG. **8** defines a so-called security boundary for the components of the access mechanism **114**. Any components required for tamper detection (tamper detect mechanism **169**) are also included as part of the access mechanism **114**. Tamper detect mechanism **169** is connected in some appropriate manner to processing unit **154**, energy source **166**, and non-volatile memory **160**.

This invention employs a combination of physical self-protection measures coupled with means for detecting that the self-protection has been circumvented or that an attempt to circumvent the self-protection measures is being or has been made. When such intrusion is detected, passive or active mechanisms can be employed to destroy data. For example, the following can occur (not necessarily in the order stated, and usually in parallel): the access mechanism **114** is made inoperative, all cryptographic keys within the mechanism, the private key and any other keys and data are destroyed (zeroized), and power may be applied to clear non-volatile memory **160** and then is removed, resulting in loss of all data stored in volatile memory **158** so as to deny access to decryption keys as well as to any cleartext in those memories. As noted above, several operations can be accommodated or performed simultaneously when tampering is detected. This can be done by hardware circuits. Based on risk assessment and the availability of particular technology, other implementations may be selected.

Tamper detection allows the access mechanism **114** to ensure that all internal data (both the system's data and any user data) are destroyed before any tamperer can obtain them.

One way to deny access to the data within access mechanism **114** is to package all of the components within a physical case which defines the area which is excluded from user access. As an example, a typical portable lap-top computer meets the requirement of having all components within the same physical package or case. Detection that the case has been opened is straightforward and well known.

As an alternative embodiment of the access mechanism **114**, the components of the access mechanism **114** can be used as a co-processor of another processor or computer. In this case, as shown in FIG. **9**, the access mechanism **114** communicates with the other computer **170** via a communications channel **172**. The co-processor can be implemented as a circuit board and is designed to be plugged into the bus **172** on the main board (that is, the mother board or planar board) of the other computer **170**. In that case, the computer **170** will operate normally unless it needs to access controlled data, at which time it will pass control to the access mechanism **114**.

The degrees of protection used in the access mechanism (for example, tamper-detect features) and the cryptographic tools employed will depend on the nature of the data to be protected as well as the user environment.

Several techniques for physically secure coprocessor packaging are described by Yee (Yee, B., *Using Secure Coprocessors*, Carnegie Mellon University, School of Computer Science, CMU-CS-94-149, 1994 (also available Defense Technical Information Center as AD-A281 255)). In Yee, physical protection is described as a tamper-detecting enclosure. The only authorized way through the enclosure is through a coprocessor-controlled interface. Attempts to violate physical protection in order to gain access to the components of the coprocessor module will be detected and appropriate action taken. For example, detection of attack results in erasure of non-volatile memory before attackers can penetrate far enough to disable the sensors or read memory contents.

Any known form of tamper protection and detection can be used, as long as it functions to destroy the data as required.

Any data which are to be sent out of the security boundary **167** are under the control of the access mechanism **114**. All I/O requests and interrupts are handled by the access mechanism **114**.

All communication between the components of the access mechanism **114** and the enclosed hard disk **162** is encrypted. Therefore, if the hard disk is removed from the mechanism, any data stored thereon will be inaccessible without the appropriate keys. The encryption of the data stored on the hard disk can use cryptographic keys generated within the access mechanism and which are never known outside of the mechanism. In this way, when tampering is detected, the cryptographic keys will be lost.

In general, within the system, the data are encrypted on any non-volatile storage devices so that they remain unavailable in the case of tampering. Unencrypted data are only present within the access mechanism **114** inside the security boundary **167** in components where the data can be destroyed when tampering with the access mechanism **114** is detected.

With reference to FIGS. **8** and **9**, the access mechanism **114** is also connected via insecure channels **174** and **176** and bus **177** to various controlled or uncontrolled display or output devices such as described above. This allows the system to communicate with uncontrolled devices (so-called standard devices) as well as networks, within the context of the rules/permission list. (Interaction with these controlled devices is described in detail below.) All communications on the insecure channels **174** and **176** and on bus **177** is encrypted by the access mechanism **114** (and by the authoring mechanism **112**), and the controlled output devices **178** and **180** must have suitable processing capabilities within them (including an access mechanism **114**) to decrypt and process data which they receive. The display or output devices used will depend on the application and the type of data, and include, but are not limited to, printers, video display monitors, audio output devices, and the like.

The embodiment shown in FIG. **9** can also include other standard devices (connected to bus **177**) such as, for example, standard printer **181**, floppy disk **185**, modem **187** and the like.

The Accessing Operation

When a user **104** obtains packaged data **108** (or **150**) from a distributor **102**, the user can then access the data according to the rules provided therewith or provided separately. Data access is supported by the access mechanism **114** and is described with reference to FIGS. **8**, **9** and **10**(*a*), where FIG. **10**(*a*) is a flow chart of the data access using the access mechanisms shown in FIGS. **8** and **9**.

Note initially that, depending on the type of data to be accessed and viewed, as well as the rules, the viewing process may or may not be interactive. For example, if a user is accessing a textual document, the user may choose to access only selected portions of that document, the choice being made by viewing an index of the document. On the other hand, if a user is accessing a movie, the viewing may be continuous (if the rules do not allow a user to re-watch portions of the movie without additional payment). The access and viewing process is described here for an interactive case, since non-interactive access can be considered access with a single ("start-viewing") interaction.

Note further that initiation of the access mechanism activates monitoring for interrupts and polling by the access mechanism **114**. A user may also implicitly invoke the access mechanism by accessing an object (data) protected by the system. This invocation also activates monitoring for interrupts and polling.

The following discussion assumes, without loss of generality, that the data are being accessed by an application via an insecure operating system (OS) which invokes the access mechanism **114**. The intent is to show the manner in which controlled access of the data takes place. In some foreseen environments, the operating system will be little more than a simple run-time system or there will be only one program running at all times. For example, in a video cassette recorder and playback machine (VCR), a single control program may be running at all times to control the VCR's operations. In this case, this control program is considered the application, and all access to controlled data is initiated by the control program which invokes the access mechanism **114**.

To initiate an input access to a data element, a user must request the operating system to read such data into memory from an I/O device. Initiating I/O gives control to the access mechanism **114**.

For input access to an input data element, the access mechanism **114** first determines whether the dataset containing the data element is already open (step **S1000**). If the dataset is not already open, it is opened (step **S1001**). Once opened, it is determined whether or not the dataset is protected (step **S1002**). Note that the data being accessed may or may not be part of packaged data. In some embodiments the access mechanism **114** can maintain a record of which open datasets are protected.

If it is determined that the dataset is not protected (step **S1002**), then control returns to the invoking process (step **S1006**). On the other hand, if the dataset is protected (step **S1002**) then it is determined whether or not the rules for this dataset are useable (present, available and valid) (step **S1004**). (The process of determining whether the rules are useable, i.e., step **S1004** is described below with reference to FIG. **11**.)

If the rules are determined to be useable (step **S1004**) then it is determined whether the data element being accessed is different from the most recently accessed data element (step **S1008**). If so, the data element is opened (step **S1010**) (otherwise the data element is already opened and available).

Next it is determined whether or not the data element is protected (step **S1012**). If the data element is not protected then control returns to the invoking process (step **S1006**). Otherwise, it is determined whether or not access is permitted (according to the rules) (step **S1014**). If no access to the data element is permitted then an access denial operation is performed (step **S1016**). For example, depending on the rules, the access mechanism **114** could either return to the invoking process (e.g., the operating system) or abort or perform some other operation. Following the access denial operation (step **S1016**), control returns to the invoking process (step **S1006**).

If access to the data element is permitted (step **S1014**), then the data element is made available, consistent with the rules, (step **S1018**) and control returns to the invoking process (step **S1006**).

If, in step **S1004**, it is determined that the rules are not useable, then an access denial operation is performed (step **S1016**), following which control returns to the invoking process (step **S1006**).

In some embodiments and/or uses of the system, the system obtains and sets up for enforcement all of the rules in the encrypted rules **124** prior to any data access or selection. In other embodiments and/or uses, rules are set up or interrogated for enforcement as needed. Depending on the type of the data and the intended application, a minimal set of global rules (governing any or all access to the data) is typically set up prior to any data access. Accordingly, the enforcement of some of the rules is set up when the package is obtained, prior-to any user access.

In some embodiments some of the required rules may not actually be provided, but are indicated by reference. In those cases, the referenced rules must be obtained when needed before data processing can continue.

Once the appropriate rules, if any, are set up (stored within the access mechanism **114**), and the access mechanism is ready to enforce them, then, according to the rules, the user can access an element of the data.

The operating system is notified of the termination (normal or otherwise) of each program so that it may close any files opened by the program. Because it is possible that multiple programs may be executing at the same time, the system will remain in a protected state (if any protected data has been accessed) until all active programs conclude their execution. At that time all protected data in addressable memory are destroyed, and all rules/permission lists of files that have been created are updated, all files are closed and system status flags are reset.

Whenever a user wishes to access protected data, the access mechanism **114** may determine that the rules are not yet available for determination of whether or not to allow that access. Three possibilities exist regarding the presence of the rules.

1. The rules are packaged with the data.
2. The rules are not packaged with the data but are already present in the access mechanism **114** (i.e., in memory). This situation occurs if, for example, the user loaded a disk containing the rules and then the access mechanism **114**, upon receiving the interrupt announcing the disk's presence, read the first record, recognized it as rules and decrypted them, storing them for later use. (Reading a disk's contents in advance of any actual use is presently done, for example, by some virus checking programs.) If the implementor chose not to respond to interrupts when a device is loaded, then, when rules are required, the access mechanism **114** checks all "ready" devices and inputs those rules that are present. This covers the case where the rules are present on the hard disk.
3. The rules are not present. That is, the rules are not packaged with the data and do not reside on any device attached to the system. In this case, the access mechanism **114** notifies the user that the rules are required. The user responds by either:
   (a) indicating that the rules are not available (in which case the access mechanism **114** denies permission to the program); or
   (b) loading the rules (in which case the access mechanism **114** confirms their identity and continues). If the access mechanism is unable to confirm their identity, it can reissue a request for the rules.

With reference to FIG. **11**, first the access mechanism **114** checks to determine whether or not the rules are already determined useable (step **S1100**). If so, the process returns a "success" indication to the invoking process (step **S1102**).

If the rules have not already been determined to be useable (step **S1100**), then the rules are located. First it is determined whether or not the rules are packaged with the data (step **S1104**). If so, the rules are made available (by decrypting them, if needed) (step **S1106**). If the rules are successfully made available (e.g., decryption succeeds) (step

S1108), then the rules are checked for integrity (step **S1110**). If the rules pass an integrity check, then a "success" indication is returned to the invoking process (step **S1112**), otherwise a "fail" indication is returned (step **S1127**).

If the rules are not packaged with the data (step **S1104**), then the access mechanism **114**, determines whether the rules are on a device attached to the access mechanism **114** (steps **S1116–S1118**). If the rules are not found on any device, then the user is asked to provide the rules (step **S1114**). At that time the user can abort the process (step **S1120**), in which case a "fail" indication is returned to the invoking process (step **S1127**). If the user chooses not to abort but to provide rules, those rules are read (step **S1122**) and, if they are a correct set of rules (step **S1124**), made available (step **S1106**). If the rules are not a correct set of rules (step **S1124**), then the user is informed (step **S1126**) and is prompted again for the rules (step **S1114**).

Regardless of whether or not the rules are provided with the packaged data, once the rules have been decrypted they are stored in the access mechanism **114**.

The process of executing an application to access the data according to the stored rules is described with reference to the flow chart shown in FIG. **12**. For each data access operation to be performed by the application, first the operation is identified (step **S1200**) and the rules are checked (step **S1202**) to determine whether that operation is permitted (step **S1204**).

If it is determined (step **S1204**) that the operation is not permitted by the rules, a "failure" return-code is set (step **S1206**) and control is returned to the caller (operating system) (step **S1208**). On the other hand, if the operation is permitted (step **S1204**) then, if payment is determined to be acceptable (step **S1210**), then processing continues. (Payment is discussed further below.) If payment is determined to be unacceptable (step **S1210**), a "failure" return-code is set and control returns to the invoking application (steps **S1206** and **1208**).

If payment is determined to be acceptable (step **S1210**), then it is determined whether or not the rules apply any restrictions on the data (step **S1212**) (for example, whether or not the rules restrict the output format or amount of the data in some way). If it is determined that the rules restrict the data then the restriction is enforced (step **S1214**) and the I/O is performed based on the restriction (step **S1216**), otherwise the I/O is performed without restriction (step **S1216**).

After performing I/O (step **S1216**), a "successful" return code is set (step **S1218**), and control returns to the invoking application.

The Writing Operation

The process of writing data is described here with reference to FIG. **10**(*b*). When an application attempts to write to a dataset, control is passed to the access mechanism **114** which opens the dataset for writing if it is not already open (steps **S1020**, **S1022**). Once opened, it is determined whether or not the dataset is to be protected (step **S1024**). The dataset (output file) would be protected if, for example, a protected dataset has been opened since the last time the access mechanism **114** cleared its memory or if the user indicated that output is to be protected (as when authoring a work).

Note that an output dataset may begin as unprotected and be written as unprotected (i.e., in the form it would have on a machine which does not have an access mechanism **114**) and later additions to the dataset may require protection and therefore be written in the appropriate format. The transition between unprotected/protected data in a dataset are discussed below.

If the dataset is not to be protected (step **S1024**), control returns to the invoking process which writes the unprotected data (step **S1026**). On the other hand, if the dataset is to be protected (step **S1024**, then the rules are checked to determine whether or not output access is permitted (step **S1028**). If output access is not permitted, a denial operation is performed (step **S1030**). For example, depending on the rules, as part of this denial operation the access mechanism **114** could destroy the output data allowing randomized data to be written in their stead, could abort the function, or could abort the job. If access is permitted (step **S1028**), it is then determined whether a new data element is about to be written or whether new rules have been incorporated since the last write (step **S1032**). If either is the case, the rules are written (step **S1034**). After writing the rules (step **S1034**), or if neither was the case (step **S1032**), the data are encrypted if the rules so require (step **S1036**), and control returns to the invoking process (step **S1026**) where the (possibly encrypted) data are written.

Compatibility Issues

A protected dataset (packaged data) read by a system which does not employ an access mechanism **114** according to the present invention (or a dataset read by a system in non-protected mode) will be treated as data without any decryption taking place (by an access mechanism). In such a system, protected data elements will not be available to the user. This allows datasets (packaged data) freely to be copied and transmitted. Recipients will need to obtain any needed permission lists (rules) prior to being able to read the encrypted data in such datasets.

A non-protected (e.g., legacy) dataset (read using a system employing an access mechanism **114**) that is treated as a protected dataset would require that rules be present before it would be accessed. The probability of such a mis-identification may be made vanishingly small, e.g., by computing a hash function of the data.

The user can be provided the opportunity to indicate that the dataset should be treated as unprotected. In order to do this, the access process described above with reference to FIGS. **10**(*a*) and **11** allows a user to override the decision made in step **S1002** as to whether or not the dataset is protected. Note that if a user incorrectly indicates that a protected dataset is unprotected, no access to the data would be available other than in encrypted (unusable) form.

Tamper Detection

If and when tampering is detected, the access mechanism **114** performs at least the following operations illustrated in FIG. **13**. The cryptographic variables (e.g., keys) are destroyed (step **S1305**), all rules are destroyed (step **S1302**), all cleartext (un-encrypted) information is destroyed (step **S1300**), all files are closed (step **S1304**), and the device is otherwise deactivated (step **S1306**). While these operations are described sequentially, in preferred embodiments they occur simultaneously or in some concurrent or parallel order, as shown in FIG. **13**. If some order must be imposed on these operations, the first priority is to erase the cryptographic variables (step **S1305**).

Operational Considerations

Certain operational procedures may also be important to maintaining the protections and controls inherent in the present invention. Specific operational procedures may be employed to prevent equipment being built that would operate with an access mechanism according to the present invention and that also contained methods for circumventing the protections and controls in the access mechanism.

These operational procedures involve inspection, analysis, testing, and perhaps other procedures followed by certification of authorized access mechanism implementations. The inspection might include design analysis and physical chip inspection. Upon successful inspection, a cryptographically sealed certificate is stored within the protection perimeter. Note that this certificate is one of the data items that is destroyed upon detection of tampering. The certificate is issued by an authorized Certification Authority (CA) and includes therein a decryption key issued by that CA.

In some preferred embodiments, the rule-encrypting key K_R is encrypted using the encryption key corresponding to the decryption key included in the certificate in each device. Then, in order to obtain K_R within the device, the device must have the decryption key which was stored in the certificate by the CA.

Payment

In our market economy, producers and distributors of goods and services expect to be compensated. Intellectual property producers and distributors are no exception. The needs of commerce have been a primary factor in the evolution of information technology throughout history. Many of today's information infrastructure activities also deal with billing and payment.

Existing payment mechanisms either assume that the parties will at some time be in each other's physical presence or that there will be a sufficient delay in the payment process for frauds, overdrafts, and other undesirable conditions to be identified and corrected. Many of these payment mechanisms have already begun to adapt in response to the conduct of business over networks. Entirely new forms of electronic payment are evolving.

The following is a representative (but not definitive) list of electronic payment systems (some of the following names are trademarks): Anonymous Internet Mercantile Protocol; "BITBUX" from "MICROSOFT" and "VISA"; CARI (Collect All Relevant Information) the Internet Voice Robot, uses virtual credit cards to provide secure transactions from the Web; "CHECKFREE" plans for expanding the way commerce is conducted on the Internet; "COMMERCENET" secure commerce on the Internet based on Secure HTTP; "CYBERCASH"; "DIGICASH"; "DOWNTOWN ANYWHERE" has a system using account numbers, and personal payment passwords; First Bank of Internet (FBOI); First Virtual Internet Payment System allows real payment on the Internet; IkP, A Family of Secure Payment Protocols from IBM; Internet Banking White Paper from WebTech; NetBill Electronic Commerce Project; "NetCash"; "NetCheque"; "NetChex"; "NetMarket"; "Netscape Communications Netsite Commerce Server" and "Netscape Navigator"; "NexusBucks"; "Open Market"; Security First Network Bank is an Internet Savings Bank; SNPP: A Simple Network Payment Protocol; Sun Internet Commerce Group; Virtual Bank of the Internet.

Some electronic payment systems operate in real time by communicating through the Internet or direct dial. Others employ a prepaid balance which is debited against merchant credits, with periodic batch updating and transmission.

It is envisioned that embodiments of the present invention will employ an appropriate payment mechanism such as are well known in the art. Accordingly, the actual payment mechanism is not specified.

Rules and Policies

The rules (provided together with or separately from the packaged data) embody the data owner's control policies with respect to a user's access rights to the data.

The present invention permits the owner of intellectual property to realize a gain by selling or licensing various levels of access rights to the property and then ensuring that access beyond those rights is not obtained. The present invention ensures that only such qualities and quantities of access as released by the owner (generally, in exchange for payment) are allowed.

The rules are preferably embodied in a permission list. An example of permissions in such a list is shown in FIG. **3**, and was described above.

While the rules allowed are open ended, an example set of rules (access control parameters) is given below. Access control parameters may be combined to provide varying sets of capabilities and to implement the enforcement of various policies. Some parameters are independent of any other parameters; some parameters are mutually exclusive; and other parameters must be used in combination to define fully the actions to be allowed or disallowed.

No Restriction

This would be the status if no restrictions were placed on the associated data. If this parameter is explicitly stated it overrides any contradictory parameter that may also be present. The data may be read, printed, executed, modified and copied.

No Modify

The associated data may not be edited or changed.

No Copy

The data may not be copied and a derivative work may not be made from the data.

No Execute

The data may not be executed.

No Print

The data may not be printed.

Print With Restriction of Type n

If the user prints after accessing the data, a simulated watermark will be printed as background or a header and/or footer will be placed on each page. The numeral n specifies the specific restriction to be applied, e.g., standard watermark (such as "do not copy"), personal (watermark such as "printed for name of user"), standard header/footer (such as "Company Name Confidential"), or personal header footer (such as "Printed for name of user").

No Access

Any user access, including an attempt to execute, will retrieve only encrypted data (ciphertext). This is the default case when there are no rules associated with data or the rules are corrupted.

No Child Access

Unless the user has been identified as an adult (for example by use of a password or a token) access will not be allowed for items identified as "adult material."

Access Cost=(Unit, Price)

Each time a unit of data (e.g., book, volume, chapter, page, paragraph, word, map, record, song, image, kilobyte, etc.) is opened, a cost of price is incurred.

Print Cost=(Unit, Price)

Each time a unit (e.g., page, file, image, etc.) is printed, a cost of price is incurred.

Copy/Transmit Cost=(Unit, Price)

Each time a unit (e.g., volume, file, record, page, kilobyte, image, etc.) is output, a cost of price is incurred.

Execute Only

The user may execute a program but may not read, print, modify or copy it. This rule protects against disclosure of an algorithm.

A permission list consists of rules governing the qualities and quantities of access made available by the owner to a particular user or group or class of users, and defines those ways in which the user may (and may not) interact with the owner's data/information. An encrypted permission list (for example, encrypted rules **124** in FIG. **2**) is made available by the owner to the user, generally in exchange for fees (in the commercial domain) (for example, payment **110** in FIG. **1**). The system denies the user direct access to manipulate the permission list, although in some cases it may allow the user to view the permission list. (The permission list may include rules governing access to the permission list itself). Use of a permission list may be limited to a particular computer system, a particular token (such as a smart card), a user-supplied password, or any combination of these or other items.

At the discretion of the intellectual property (data) owner, a permission list may also be valid for future releases of the data. This allows, for example, a software owner to plan for future releases that resolve problems discovered in an initial software release. In this example, the user of a particular version of a program, for instance, Version 6, might be allowed to use a subsequent version of the program, version 6.1, without further payment and without needing to obtain a new permission list or license. One who had not already licensed Program Version **6** would be required to purchase a new permission list/license in order to use Program Version 6.1.

A permission list may authorize and permit the user of intellectual property to create a derivative product for which the original owner may or may not have rights. In the case of a derivative product for which the owner of the original intellectual property has no rights, the owner of the derivative intellectual property can unilaterally issue a permission list governing use of that intellectual property.

Program execution occurs when a computer device follows a series of steps, or instructions, expressed in some symbology. The program may be linear, with one step always following its predecessor without variation, or the program may involve branching based on comparison of variables related to internal or external events and status. In the field of computer science a distinction is sometimes made according to the time at which the instructions comprising the program are translated into the computer's machine language in order to control the operation of the computer. Accordingly, terms such as assembly, compilation, and interpretation are used. This distinction is not important with respect to the present invention. The term execution is used herein to refer to all forms of program execution.

Controlling Primary Distribution

As noted above, digital information is transmitted openly. Accordingly, the data are typically distributed in an encrypted form.

Enforcing an Authorized User List

In some cases, it is useful to have a rule which controls access to data for certain specific users or classes of users. For example, data may only be accessible to people over the age of eighteen, or to people having a rank greater than or equal to that of captain, or to managers have a security clearance greater than top-secret. In these cases, each user can be provided with a separate set of rules for that specific user. In other words, each user can be provided with a unique set of rules. However, if the status of a user changes, then the rules for that user have to be changed. Accordingly, it is useful and convenient to have the rules be parameterized based on the status of the user and then have the user's status provided to the access mechanism **114** in a secure fashion.

The invention can be used in combination with software and other identification technology (for example, biometric sensors) to limit data access to users that possess an appropriate physical or logical token (for example, a dongle or password), or personal characteristic (for example, a fingerprint pattern). The secure hardware (via tamper detection) eliminates the potential for modifying and subverting the identification software.

An embodiment having such a configuration is shown in FIG. **14**, wherein the access mechanism **114** is connected to an external secure device **182** in order to obtain the user's status. Channel **183**, connecting the secure device **182** and the access mechanism **114** is preferably a secure channel (within the security boundary **167**), however, if it is insecure, the device **182** must send information to the access mechanism **114** in a protected (e.g., encrypted) manner.

Controlling Access and Use

The invention can restrict the qualities or quantities of access to data in any manner that can be calculated or enumerated. A non-exhaustive, representative set of examples is given below.

Access Control Qualities

(a) Local Display (for example, display of data on the computer's monitor).

(b) Printing (i.e., fixation in a form intelligible to a person).

(c) Copying (i.e., fixation on an electronic medium such as a disk or tape).

(d) Transmission (see below regarding controlling secondary distribution).

(e) Modification (i.e., changes to a copy of the primary distribution).

Access Control Quantities

(a) Number of read-accesses (where "read access" refers to any kind of examination or retrieval of data/information).

(b) Size of read-access.

(c) Expiration date.

(d) Intensity of access (number/total volume of read-accesses in a unit of time).

(e) Resolution of access (for example, in the context of a map this would be the maximum scale allowed; for sensor data this would be the precision (number of bits) returned to the user).

(f) Delay (Accesses are permitted to data after a delay of n time units. This allows different user groups to view the same dataset with different results to queries. For example, a stock broker would be able to view the latest data, while a customer, paying less for the service, might receive data that are delayed by 15 minutes.)

Access Control Granularity

The above access control policies can be applied differently to different portions of the intellectual property. For example, a document's chapters might be controlled at different levels of quantity and quality; is a map's information might be controlled differently at different latitudes and longitudes; portions of an image may be restricted in availability, resolution, and the like.

Controlling Secondary Distribution

The invention provides absolute control of secondary distribution of data (for example, preventing or restricting potential use).

Transmission of (an unencrypted copy of) the primary distribution data (either to a network or to an output device such as a tape or disk) can only be effected when the system, acting under the rules embodied in the owner's permission list, allows external output. Denial of permission to transmit an unencrypted copy may result in no output or may result in transmission of an encrypted copy (for which the recipient must then negotiate permissions in order to use). Alternately, denial of permission to transmit may result in the transmission of random data, thereby denying the user knowledge of whether or not encrypted data was transferred.

Since all storage of data on internal non-volatile memory devices (for example, disks, flash memory, and the like) is encrypted, this ensures that a physical attack on the system will not result in compromise of plaintext.

Controlling Printing or Display

Printing or display of data is controlled in a manner similar to that used for controlling secondary distribution. One option is to disallow the ability to send particular information to a printer or display. If printing or display is allowed, the data stream to the output device is encrypted to ensure that an unauthorized user cannot intercept data sent to an external printer or display (that is, to a printer or display outside the tamper-detect protected enclosure). This necessitates that the receiving device contain a decryption subsystem. Thus, as shown in FIG. **8**, data from access mechanism **114** via I/O controller **165** to either the controlled printer **178** or the controlled display **180** is encrypted on channels **174** and **176**, respectively.

As discussed above when addressing the threat of capture of the output signal, an encryption mechanism is used for protecting data transfers to printer or display so that, if the data owner wishes, printing or display may be restricted to a specific printer or display device.

Instead of disallowing printing or display, these functions may be allowed with limitations as imposed by the owner. For example, output might contain a header/footer on each page indicating the identity of the authorized user; a watermark might be printed in the background; or other identifying material might be placed on each image. Of course, the data stream would be encrypted (as above) to prevent interception.

Document marking and identification techniques can be used to discourage the illicit copying of documents distributed in either paper or electronic form. The exact form of printer characters as well as line and word shifting have been used for document marking and identification ("Document Marking and Identification using both Line and Word Shifting," Low, S. H., et al. 1995 *INFOCOM Proceedings, IEEE*, pp. 853-, 1995).

One of the major technical and economic challenges faced by electronic publishing is that of preventing individuals from easily copying and illegally or without authorization distributing electronic documents. Cryptographic protocols used to discourage the distribution of illicit electronic copies are described in "Copyright Protection for Electronic Publishing over Computer Networks," Choudhury, A. K., et al., *IEEE Network*, pp. 12–20, May–June 1995.

Preferably, each controlled peripheral device (e.g., controlled printer **178** or display **180**) is provided with an access mechanism which allows the device to process data it receives. This allows the data being sent to a controlled peripheral device from a system using an access mechanism to be treated as either a copy of data or a derivative work that is being sent to another user (that happens to be a peripheral). In other words, if a peripheral device contains an access mechanism, the data sent to the device can be packaged data. Using this approach, requires that the receiving access mechanism (the peripheral's access mechanism) may include the rules (permission list(s)) in order to obtain the key needed to decrypt the data in order to print or display them (or do whatever the peripheral does with data). If no permission list is included and the data are encrypted by the printer's public key, the printer's access mechanism decrypts the data and prints them (just as they would have been printed had the unencrypted data stream been received by a standard printer).

The access mechanism in the controlled peripheral device need not be a full system whenever the peripheral device is limited in function, for example, to only printing or displaying data. The peripheral and its access mechanism subsystem must be in a tamper-detecting enclosure.

As noted, it is envisioned that a computer or other device equipped with an access mechanism will be used with a controlled output device (printer or display) so equipped. If the data owner allows (via the rules) output (e.g., printing) to a controlled output device (e.g., printer) (equipped with an access mechanism), then there are two possibilities. The access mechanism in the user's computer can process any required payment and send the data, encrypted with the device's public key, to the printer or display for output. Alternately, the access mechanism processes the data as a derivative work (discussed below), packaging rules with the data, and the output device is responsible for separate payment (for example, allowing retention and multiple copies).

In order to limit the number of copies output, a short time window is included in the rules so that the recipient cannot capture (record) the file and replay it multiple times to the output device. Additionally, the access mechanism in the output device can contain a relatively small non-volatile memory that would hold the checksum of a file that is not to be output again for a certain time period, say, for 15 minutes from the first output (and an output permission list in the rules would specify "n copies, only valid for 15 minutes from x to x+15").

In the case of standard output devices (non-controlled, i.e., without access mechanisms), data are provided unencrypted (to the extent that the rules permit and payment has been provided).

Controlling Distributions of Derivative Works

In many application environments where intellectual property is created it is common to include extracts from other intellectual property. Such environments include writing scholarly papers, reviews, regulations, etc. The intellectual property containing the extract is a so-called derivative work. The intellectual property from which the extract was copied is called the parent work.

This invention controls the distribution of derivative works (that is, works created using information owned by another). Transmission of (an unencrypted copy of) a derivative work (to a network, to an output device such as a tape or disk, or to a printer or display device or the like) can only be effected when the system, acting under the rules embodied in permission lists created by each of the owners of any intellectual properties used in the derivative work, allows external output. Denial of permission to transmit an unencrypted copy may result in no output or may result in transmission of an encrypted copy (or, as noted above, may result in the transmission of random data). Use of an encrypted copy of a derivative work will, in general, require permissions from the owners of the derivative work as well as of the original works. The permission list associated with a work is incorporated into the permission list of any derivative work, either directly or by reference. License fees and restrictions imposed by the owner of a work are inherited by any derivative works. An n-th generation derivative work inherits the license fees and restrictions of each of its n−1 ancestors. If permission lists (rules) are incorporated by reference, the access mechanism ensures that the referenced permission lists (rules) are present (or it will deny access).

For example, if printing of an original work requires a watermark, then printing of any derivative work (if allowed at all) will require a watermark. This monotonicity/cascading of restrictions (i.e., each generation of a work must be at least as restricted as the prior generation) ensures that a derivative work that is only trivially changed from the original does not escape restrictions imposed on the original.

Creation of a derivative work for subsequent distribution requires an distributor **190** similar to distributor **102** shown in FIGS. **1** and **5**. However, derivative work distributor **190** (shown in FIG. **15**) includes an access mechanism **114** and can process, as input data, packaged data **108***a*. The output produced by distributor **190** is packaged data **108***b* which includes any rules (or references to rules) required by data which is derived from the input packaged data **108***a*. The access mechanism **114** within distributor **190** incorporates a global rule which enforces the distribution of rules with derivative works.

As noted earlier, the difference between the embodiments of the distributors **102** and **190**, shown in FIGS. **1** and **15**, respectively, is that the distributor **102** shown in FIG. **1** does not include an access mechanism **114**. Accordingly, the distributor **102** deals only with newly created data (that is, with non-derivative data). The embodiment shown in FIG. **15** includes that of FIG. **1**, and can also deal with input of protected data (previously packaged by a distributor). The embodiment of the system shown in FIG. **1** can be implemented purely in software, whereas the embodiment shown in FIG. **15** requires some hardware implementation.

It is envisioned that a standard computer, equipped with an access mechanism **114** will function as an authoring/distribution system. This allows all computer users to become authors and to incorporate previously published material into derivative works.

The rules associated with the parent work determine whether creation of derivative intellectual property is permitted, as well as the inheritance rules for incorporating the rules of the parent into the derivative work. Note that the rules derived from the parent apply only to the extract and that these rules applying to the extract need not be identical to the rules of the parent. The rules applying to the extract are specified by the owner of the parent, not by the creator of the derivative work.

For example, the rules applying to the extract might require payment to the owner of the parent for use of the derivative work containing the extract. If the creator of the derivative also required payment, the user of the derivative would make payments to two owners for use of the derivative. In an automated system the details of such multiple payments would be invisible to a user.

This invention enables such payment arrangements that would otherwise be prohibitively difficult and complex.

Another example relates to integrity and moral rights of the owner of the parent. The owner might wish to ensure that an extract was made without alteration or deletion, or that certain related information were included (for example, to prevent the extract from being taken out of context).

Data extracted from the parent comes with rules already attached or associated. These rules propagate into the derivative, but are applicable only to the extract. Extracts from the same parent may or may not share rules. Extracts from multiple parents may result in multiple rules applying to different extracts. As noted, a derivative work may contain references to data and rules rather than the actual data and rules. For certain commercial products it may be desirable to have the final packaged data **108*b*** be fully self-contained. Accordingly, the packaged data **108*b*** output from this distributor **190** may require further processing in order to optimize it for commercial distribution. Such optimization might include, for example, obtaining and including copies of all rules and data referenced in the package.

Extract Authentication

Digital signatures authenticate digital information by providing proof that information received is precisely that which was sent, with no changes. This system provides a similar capability to authenticate extracts (quotes) of information.

Application environments, such as providing a legal trail of evidence or authenticating that a quotation is accurate, are enhanced by the ability to prove that the information has not been subject to unauthorized alteration.

Authenticated extraction is implemented by creating an extraction editor, that runs in the access mechanism **114**. This extraction editor, possibly under human direction, can extract selected text but is unable to change the extract. When extraction is complete, the access mechanism **114** digitally signs the extract with a digital signature. This digital signature includes identification of the specific computer in which the access mechanism **114** is executing as well as identification of the specific extraction editor used.

The extraction editor can, optionally, be permitted or required to insert ellipsis to indicate deletions, and certain specified insertions, such as, for example, "[sic]," might be allowed.

In another embodiment, a so-called hyperlink can be used in newly created data to indicate the insertion location of a quotation. When an output operation is performed, the access mechanism **114** creates a separate quotation, with its own checksum and digital signature. Any recipient of data containing the hyperlink can verify that the contents of the hyperlink were captured by access mechanism **114** and delivered unchanged.

Controlling Use of Executable Software

Control of Primary Distributions

The invention enables the creator of executable software to restrict the use of the software to only those who have acquired permissions for various of its capabilities. Executable software will be distributed in encrypted form, externally treated as data, as described above. In general, execution of a program can be controlled for multiple purposes in a number of ways. Purchase of a license to execute software can be evidenced by a cryptographically protected certificate which is decrypted internally by the access mechanism **114**. The executable software can check for the presence of the certificate, or for permission keys or other information contained in the certificate, once or many times during execution. Since the algorithm embodied in an executable program may be valuable intellectual property, the access mechanism **114** can prevent a licensee from reading, copying, or modifying unencrypted executable code. In order to prevent disclosure of the unencrypted executable code, it is kept wholly within the security perimeter of the access mechanism **114** for execution.

Elimination of the Distributor (Middleman)

The invention enables the executable software owner to make copies easily available on a network server in encrypted form. Users may download the executable software and then separately purchase the rights to utilize the executable software. Thus, a standard purchase of software may be accomplished electronically, dealing with the owner's electronic commerce system. Thereby, the entire process of acquiring the executable software package and then

purchasing the rights to use it may be effected without going through a distributor.

Offering discounted upgrades to software licensees is also simplified. When a licensee claims eligibility for a discounted upgrade the executable software owner can check the record of purchase of rights for the prior version of the product. Once again, the entire process can be automated.

Simplification of Configuration Management

The executable software owner can elect to make available on a network server product improvements that operate with existing permission lists, thus immediately releasing product improvements and fixes.

Multiple levels of product capability can be incorporated into a single release and can be selectively enabled by different permission lists. The tailoring of different distributions, with differing capabilities is no longer necessary.

Active Control of Capability of Executable Software

The invention's control of distribution of data or information (that are not executable software) may be characterized as passive or transparent in that no changes are required in the data or information for them to be protected. The permission list that controls their use may be separately created, packaged, and supplied.

The control of primary distribution of data or information as well as the secondary distribution or distribution of modifications (derivatives) of data or information is passive. However, the invention's control of executable software capability is active and requires that the executable software developer use the programming interface provided by the system. At each point where the developer requires authorization, the executable software requests-a permission-check. As a result, the process of FIG. **16** is performed. If the requisite authorization is received, the function of the software is performed. If authorization is denied, an alternative action is chosen. The system may itself take certain actions including, for example, terminating a program or erasing data, when authorization is denied. As executable software is distributed in encrypted form, it can only be decrypted and executed (used) on a machine employing the access mechanism of the present invention.

With reference to FIG. **16**, first the operation is identified (step **S1600**) and the rules are checked (step **S1602**). Next it is determined whether the rules permit the operation (step **S1604**). If the operation is not permitted (or it is permitted but payment is not acceptable (step **S1606**)), then it is determined whether any system action is required (step **S1608**). If no system action is required, the return code for "not allowed" is set and control is returned (step **S1610**), otherwise the system action is performed (step **S1612**) after which the return code for "not allowed" is set and control is returned (step **S1610**).

If the operation is permitted (step **S1604**) and payment is acceptable (step **S1606**), then the return code for "allowed" is set (step **S1616**).

The invention can be used to restrict the qualities or quantities of executable software execution in any manner that can be calculated or enumerated. Representative non-exhaustive examples of restrictions are given below. These restrictions may combined in any fashion.

Levels of Capability

Access to Specific Parts of Code or Features

Control of sizes or quantities that can be handled. For example, files may be allowed up to a specific size; complexity or accuracy of a solution may be limited, number of parameters or data points may be restricted, etc.

Quantitative Modifiers of Levels of Capability

Control of expiration dates, time of use, number and frequency of uses and permitted users. For example, rights to use of a file of data (whatever it contains) may expire on a certain date; access to certain data may be limited to certain times of day, days of the week or specific dates; a user may only be allowed to access certain data a specified number of times (or a specified number of times per day); or access to some data may be restricted based on the identity of the user.

Control of Secondary and Derivative Executable Software Distributions

This is handled in the same fashion as are data files, as described above.

Control of Executable Software as a Module of Other Executable Software

When protected executable software is incorporated into or used by other executable software on the system for which it was licensed, any limitations on its execution are maintained in the new context.

Restricting Use to Certified Software

The access mechanism **114** can be factory configured to restrict operation only to such software as is certified (e.g., by using a digital signature to ensure that the software was received unaltered from a certified source). Other contemplated applications include key escrow (also called "data recovery") systems (described below), systems for counting election ballots, systems for exchanging cryptographic data or algorithms, and systems for safeguarding financial, medical, or other personal data. Further, a system employing an access mechanism may be used to ensure that such software is not modified after being received or accessed for execution.

Process Control

Computer control of processes is the basis for automation and quality control in many industries. This technology extends into various specialties such as computer-aided manufacturing, control systems engineering, concurrent engineering, expert systems, intelligent sensors, just-in-time manufacturing, programmable logic controllers, robotics, robotic programming languages, and visualization techniques in engineering.

Formula, processes, procedures, and techniques may convey product differentiation, aesthetic and functional innovation, and increased cost-effectiveness. The computer programs and data involved in process control may constitute valuable intellectual property. The mechanisms of the present invention permit such data to be stored in process-control computers, transmitted to suppliers and subcontractors and otherwise employed without unauthorized disclosure, substitution, or modification.

The permissions associated with process control data may, for example, allow execution only—reading or observing the data would be prohibited. Execution may be restricted to specific equipment and to specific times. In general, the process controller is external to the equipment implementing the process. Hence, communication between the process controller and the process equipment must be cryptographically protected. Like the access mechanism in a controlled computer peripheral discussed herein, the access function in the process equipment need not be a full system whenever the peripheral device is limited and can not output data.

Key Escrow (Data Recovery) Systems

This system allows a provider of key escrow cryptographic executable software to require, by using a rule, certification that a key has been installed and deposited with a specified certification authority in order for the executable software to function. The access mechanism ensures the integrity of executable software that uses cryptographic executable software (whether or not key escrow), guarding against change or replacement.

Control of Classified Data

The invention can be used to support limitations on the (primary and secondary) distribution of data, access to data, and distribution of derivative data where the data are classified. Similarly, the execution of classified programs, or programs operating on classified data may be controlled by the system.

Ensured Issuance of Receipts

This system can be used to ensure that a receipt is issued under a number of circumstances, as demonstrated by representative examples given below. A software program (or electronic mail message) may request that a receipt be issued whenever it is loaded or executed (or when a mail message is received); a receipt may be issued when a mail message is read for the first time; or a program will not be loaded or executed (or mail opened for reading) unless the user first agrees to allow a receipt to be issued.

Ensuring Privacy

This system can be used to ensure privacy of sensitive records in a database. Examples include financial, census, medical, and political databases and the like. The system can allow inquiries that provide statistical summaries but do not reveal information about individuals. The rules would be used to limit the queries that might be posed.

Owner Control/Privileges

At the time of purchase the identity of the owner may be stored within the access mechanism. The access mechanism may allow the owner to place a global set of rules (a global permission list) in the mechanism. These global rules could control, for example, hours of access (e.g., when the computer might be operated) based on a clock within the access mechanism or an external time reference with which the access mechanism communicates; acceptable software which can be run using the access mechanism (i.e., a list of those software products that would be allowed to be used, thus enforcing a system administrator's configuration control rules); user and password lists, and the like. A user can thereby customize a particular access mechanism.

The rules may also include or specify certain programs to be run under certain conditions. For example, if the rules specify that all printed output must contain a watermark, the rules might also provide the watermark generating program. In these cases, the programs are either pre-loaded into the access mechanism **114**, or are loaded when needed. These programs will then be executed when the corresponding rules or functions are invoked. For example, various types of watermark programs can reside in the access mechanism **114**, and, depending on the rules, the appropriate one of these can be selected and executed.

Note that the data structures in FIGS. **2** and **6** depict logical organizations of the data. However, the actual physical format of the data depends on the type of the data as well as on the manner in which the data are to be used. Further, as noted above, the data package may be distributed in many ways, including networks, magnetic media, CD-ROM, semiconductor memory modules, and wireless broadcast and the like. In certain types of data distribution, e.g., continuous cable or wireless broadcast, a user may wish to begin accessing the data at an arbitrary point during its distribution. For example, if the data represent a broadcast movie which begins at 8 p.m., a particular user may only begin viewing at 8:30 p.m. In this case the user will have to initiate reception of the distribution while it is in progress.

Accordingly, as shown in FIG. **17**(*a*), in some embodiments, the packaged data are distributed in discrete packets **236** of data. The packets **236** include information **238** which enables a user to synchronize with the data distribution and further enables the user to begin accessing the data according to the rules. An example of such a packetized stream of data is shown in FIG. **17**(*b*) wherein the stream **234** consists of discrete packets **236** of data, each packet containing synchronization data **238**.

EXAMPLES

The following examples indicate some envisioned data and its packaging and rules. These examples are only intended to show some of the envisioned uses of the present invention, and are in no way intended to limit its uses.

Books

With reference to FIG. **18**(*a*), a digital book **191** consists of an abstract **192**, an index **194**, and various chapters **196**. Each chapter **196** comprises sections **198**, and each section comprises text **200** and figures **202**. The distributor can decide to package the book **191** such that the abstract **192** and the index **194** are available for browsing, but all other data are protected (encrypted). If the rules specify that the text is restricted in certain ways, then the packaged data structure **108** has the form shown in FIG. **18**(*b*), wherein encrypted body part **120** includes all chapters **196**, unencrypted body part **122** includes the abstract **192** and index **194**, and encrypted rules **124** contains the encrypted version of the rules.

Movies

With reference to FIG. **19**(*a*), a movie **204** can be made such that different parts of the movie combine to form either a trailer **206**, a G-rated version (from G-rated parts **208**), an R-rated version (formed from G-rated parts **208** and R-rated parts **210**) or an X-rated version (formed from G-rated parts **208**, R-rated parts **210** and X-rated parts **212**). The packaged data structure **108** for this movie has the form shown in FIG. **19**(*b*), wherein encrypted body part **120** includes all the G, R and X-rated parts **208–212**, unencrypted body part **122** includes the trailer **206**, and encrypted rules **124** contains the encrypted version of the age-based rules which control viewing of the various versions of the movie.

In one embodiment, as shown in FIG. **19**(*c*), a movie may be released with a main body **207** (having elements common to all three versions) and sections for each of the G, R and X-rated parts (**208**, **210**, **212**, respectively). Sections of the movie are selected from one of the rated parts, depending on the permission level (G, R or X) set. FIG. **19**(*d*) shows packaged data structure **108** for such an arrangement.

Software

With reference to FIG. **20**(*a*), a software program such as, for example, a word-processor **214** may include a controlled file access part **216**, an editor **218**, a grammar checker **220**, and other features **222**. The rules obtained by the user will govern the features of the software that may be used and the quantities of data that may be processed. The rules shown in FIG. **20**(*c*) indicate that the user may not employ the grammar checker and may operate on no more than nine files. The packaged data structure for this software (without rules) **150** is shown in FIG. **20**(*b*), wherein encrypted body part **120** includes the file access mechanism **216**, the grammar checker **220** and various other functions **222**, and unencrypted body part **122** includes the editor **218**. The encrypted rules **124** are shown separately in FIG. **20**(*c*).

Documents

With reference to FIG. **21**(*a*), a document such as a legal document **224** comprises paragraphs **226** of words **228**. In

order to limit access to non-redacted portions of the document, the rules would require blacking out all redacted words. Accordingly, the corresponding packaged data structure is shown in FIG. **21**(*b*), wherein encrypted body part **120** includes the redacted portions of the document and unencrypted body part **122** contains the non-redacted portions of the document.

Map Image Data

With reference to FIG. **22**(*a*), map image data **230** may be available at three resolutions (high, medium and low). The rules may specify that people with a security clearance of greater than "top-secret" can view the data at high resolution, and all non-military users can only view the map data at low resolution. The corresponding packaged data structure is shown in FIG. **22**(*b*), wherein encrypted body part **120** includes all data beyond low resolution (that is, those data required for medium and high resolution) and unencrypted body part **122** contains the low resolution data.

Global Positioning System (GPS) Software

With reference to FIG. **23**(*a*), GPS software includes an output routine **232** which can produce output at various degrees of accuracy. The degree of accuracy depends on the security clearance of the user. A corresponding packaged data structure is shown in FIG. **23**(*b*), wherein encrypted body part **120** includes the resolution calculation routine **232** and unencrypted body part **122** contains the other parts of the GPS software **230**.

Relationship Among Rule Sets

In some embodiments, the access mechanism may be supplied with a set of rules built-in. In such an access mechanism the built-in rules might include rules that can or cannot be overruled (made less restrictive) by rules provided with packaged data. These initial rules can perform a number of functions and implement a number of policies. As examples, the access mechanisms provided in controlled output devices can include built-in rules (that cannot be overruled). which limit the device only to being an output device; or, the access mechanism provided with a VCR or a videodisc player can include rules (that cannot be overruled) which require the device to enforce the copyright laws of the country in which the device is sold. Whether or not internal built-in rules can be overruled by rules provided externally can be specified in the internal rules themselves.

While the present invention may be used to protect intellectual property by controlling access to that property, the mechanisms discussed herein are technical in nature and are independent of any form of legal protection—a purely technological approach has been presented to controlling access to data. Indeed, the invention offers the intellectual property owner the opportunity to restrict access and use of his or her data beyond the protections that may be available in law. The protection offered by the present invention may be used to enforce rights in intellectual property whether the protection at law is categorized as copyright, trade secret, contract, or something else. The cost-benefit tradeoff of seeking protection at law must be made by those with a vested interest in the intellectual property.

Typical computer systems are implemented at various levels, each level effectively defining a different virtual machine. Generally, each level of implementation can access the levels below it. In many systems it is desirable to have each level only access the level immediately below it. In that way, various policies can be enforced.

Typically the higher level virtual machines are implemented in software and the lower level machines are implemented in hardware. However, there is no precise hardware/software boundary between levels.

With reference to FIG. **24**, for example, a computer system has a high-level application environment (level L**4**). These applications invoke (call) operating system level (L**3**) processes to perform various system functions. The OS level (L**3**) processes in turn invoke lower-level Basic Input/Output System (BIOS) machine dependent instructions as required (level L**2**). Note that application level (L**4**) programs might be permitted to bypass the OS level (L**3**) and invoke BIOS level (L**2**) processes directly, thereby avoiding any OS level (L**3**) policy checking and enforcement.

As an example, an application (executing a level L**4**) program which wishes to open a particular named file would invoke an operating system "open" procedure for that named file. The OS determines the location of the file (using, for example, an internal map between file names and locations) and then invokes a lower level (L**2**) BIOS routine to perform the actual seek to the file and the open and read. However, the application program might be permitted to bypass the operating system's "open" process and invoke the BIOS routines directly.

It is desirable to implement the access control mechanisms of the present invention at a low level, preferably at or below the BIOS level (level L**1**). This prevents users from by-passing the access control mechanisms of the invention and thereby circumventing the rule enforcement.

Thus, a system for controlling access and distribution of digital property is provided. One skilled in the art will appreciate that the present invention can be practiced by other than the described embodiments, which are presented for purposes of illustration and not limitation, and the present invention is limited only by the claims that follow.

What is claimed is:

1. A method of distributing data, the method comprising:

protecting portions of the data; and

openly distributing the protected portions of the data, whereby

each and every access to an unprotected form of the protected portions of the data is limited in accordance with rules defining access rights to the data as enforced by an access mechanism, so that unauthorized access to the protected portions of the data is not to the unprotected form of the protected portions of the data.

2. A method as in claim **1**, wherein

the protecting of portions of the data comprises encrypting the portions of the data, whereby unauthorized access to the protected data is not to the un-encrypted form of the protected data.

3. A method as in claim **2**, wherein the encrypting of portions of the data encrypts the portions of the data with a data encrypting key, the data encrypting key having a corresponding data decrypting key, the method further comprising:

encrypting the data encrypting key.

4. A method as in claim **3**, further comprising:

providing a decrypting key corresponding to the key encrypting key.

5. A method as in claim **1**, wherein the data represent at least one of software, text, numbers, graphics, audio, and video.

6. A method as in claim **1**, wherein the rules indicate which users are allowed to access the protected portions of the data, the method further comprising

allowing the user access to the unprotected form of a protected portion of the data only if the rules indicate that the user is allowed to access that portion of the data.

7. A method as in claim **1** wherein the rules indicate distribution rights of the data, the method further comprising:

allowing distribution of the unprotected form of the protected data portions only in accordance with the distribution rights indicated in the rules.

8. A method as in claim **1**, wherein the rules indicate access control rights of the user, the method further comprising:

allowing the user to access the unprotected form of the protected data portions only in accordance with the access control rights indicated in the rules.

9. A method as in claim **8**, wherein the access control rights include at least one of:

local display rights,

printing rights,

copying rights,

execution rights,

transmission rights, and

modification rights.

10. A method as in claim **1**, wherein the rules indicate access control quantities, the method further comprising:

allowing access to the unprotected form of the protected data portions only in accordance with the access control quantities indicated in the rules.

11. A method as in claim **10**, wherein the access control quantities include at least one of:

a number of allowed read-accesses to the data;

an allowable size of a read-access to the data;

an expiration date of the data;

an intensity of accesses to the data;

an allowed level of accuracy and fidelity; and

an allowed resolution of access to the data.

12. A method as in claim **1**, wherein the rules indicate payment requirements, the method further comprising:

allowing access to the unprotected form of the protected data portions only if the payment requirements indicated in the rules are satisfied.

13. A method as in claim **1**, wherein the rules relate to at least one of:

characteristics of users;

characteristics of protected data; and

environmental characteristics.

14. A method as in claim **1** wherein the rules defining access rights include at least one internal rule built in the access mechanism.

15. A method as in claim **14** wherein the at least one internal rule cannot be made less restrictive by any other rules.

16. A method as in claim **14** wherein the access mechanism is contained in a stand-alone device.

17. A method as in claim **16** wherein the stand-alone device is selected from the group consisting of: a facsimile machine, a television, a VCR, a laser printer, a telephone, a laser disk player, and a computer system.

18. A method as in claim **1**,

wherein the access mechanism is contained in a stand-alone device selected from the group comprising: a facsimile machine, a television, a VCR, a laser printer, a telephone, a laser disk player, and a computer system; and

wherein the rules defining access rights include at least one internal rule built-in to the access mechanism; and wherein the at least one internal rule comprises access control rights to the data.

19. A method as in claim **1**, further comprising:

providing a distribution rule,

wherein the rules defining access rights comprise the distribution rule and at least one internal rule built in to the access mechanism.

20. A method as in claim **19** wherein the protecting of portions of the data comprises encrypting the portions of the data using a data encrypting key having a corresponding data decrypting key, and wherein the distribution rule comprises a data decrypting key.

21. A method of distributing data for subsequent controlled use of the data by a user, the method comprising:

protecting portions of the data;

protecting rules defining access rights to the data; and

openly distributing the protected portions of the data and the protected rules, whereby

controlled access to an unprotected form of the protected portions of the data is provided only in accordance with the rules as enforced by an access mechanism, so that unauthorized access to the protected portions of the data is not to the unprotected form of the protected portions of the data.

22. A method of distributing data for subsequent controlled use of the data by a user, some of the data having access rules already associated therewith, the access rules defining access rights to the data, the method comprising:

protecting portions of the data;

providing rules defining access rights to the data;

combining the provided rules with rules previously associated with the data;

protecting the combined rules; and

openly distributing the protected portions of the data and the protected combined rules, whereby

controlled access to the unprotected form of the protected portions of the data is provided only in accordance with the combined rules as enforced by an access mechanism, so that unauthorized access to the protected portions of the data is not to the unprotected form of the protected portions of the data.

23. A method of controlling secondary distribution of data, the method comprising:

protecting portions of the data;

protecting rules defining access rights to the data;

openly providing the protected portions of the data and the protected rules to a device having an access mechanism; and

limiting transmission of the protected portions of the data from the device (a) only as protected data or (b) in accordance with the rules as enforced by the access mechanism, so that unauthorized access to the protected portions of the data is not to an unprotected form of the protected portions of the data.

24. A method of accessing openly distributed data, the method comprising:

obtaining openly distributed data having protected data portions and rules defining access rights to the protected data portions; and

limiting each and every access to an unprotected form of the protected data portions in accordance with the rules as enforced by an access mechanism, so that unauthorized access to the protected portions of the data is not to the unprotected form of the protected data portions.

25. A device for displaying images represented by data comprising protected data portions and rules defining access rights to the data, the device comprising:

means for storing the rules;

an access mechanism for accessing the data only in accordance with the rules, whereby user access to an unprotected form of the protected data portions is permitted by the access mechanism only if the rules indicate that the user is allowed to access the protected portions of the data, the access being enforced by the access mechanism; and

means for displaying the images represented by the accessed data.

26. A device as in claim **25** wherein the rules defining access rights include at least one internal rule built in the access mechanism.

27. A device as in claim **26** wherein the internal rules cannot be made less restrictive by any other rules.

28. A device as in claim **26** wherein the internal rules limit the device only to being an output device.

29. A device as in claim **26** wherein the device is selected from the group consisting of: a VCR, a laser disk player, and a computer system.

30. A device for outputting images represented by data comprising protected data portions and rules defining access rights to the data, the device comprising:

means for storing the rules;

an access mechanism for accessing the data only in accordance with the rules, whereby user access to an unprotected form of the protected data portions is permitted by the access mechanism only if the rules indicate that the user is allowed to access the protected portions of the data, the access being enforced by the access mechanism; and

means for outputting the images represented by the accessed data.

31. A device for outputting an audio signal represented by data comprising protected data portions and rules defining access rights to the data, the device comprising:

means for storing the rules;

an access mechanism for accessing the data only in accordance with the rules, whereby user access to an unprotected form of the protected data portions is permitted by the access mechanism only if the rules indicate that the user is allowed to access the protected portions of the data, the access being enforced by the access mechanism; and

means for outputting the audio signal represented by the accessed data.

32. A device for outputting an output signal based on data comprising protected data portions and rules defining access rights to the data, the device comprising:

means for storing the rules;

an access mechanism for accessing the data only in accordance with the rules, whereby user access to an unprotected form of the protected data portions is permitted by the access mechanism only if the rules indicate that the user is allowed to access the protected portions of the data, the access being enforced by the access mechanism; and

means for outputting the output signal represented by the accessed data.

33. A device for generating an output signal corresponding to data comprising protected data portions and rules defining access rights to the digital data, the device comprising:

means for storing the rules;

an access mechanism for accessing the digital data only in accordance with the rules, whereby user access to an unprotected form of the protected data portions is permitted by the access mechanism only if the rules indicate that the user is allowed to access the protected portions of the data; and

means for generating the output signal from the accessed data.

34. A device for distributing data for subsequent controlled use of the data by a user, the device comprising:

means for protecting portions of the data;

means for protecting rules defining access rights to the data; and

means providing the protected portions of the data and the protected rules;

whereby a user is provided controlled access to the data only in accordance with the rules as enforced by an access mechanism, so that unauthorized access to the protected portions of the data is not to an unprotected form of the protected portions of the data.

35. A device for distributing data for subsequent controlled use of the data by a user, some of the data having access rules already associated therewith, the access rules defining access rights to the data, the device comprising:

means for protecting portions of the data;

means for providing rules concerning access rights to the data;

means for combining the provided rules with rules previously associated with the data;

means for protecting the combined rules; and

means for providing the protected portions of the data and the protected combined rules;

whereby the user is provided controlled access to an unprotected form of the protected portions of the data only in accordance with the combined rules as enforced by an access mechanism, so that unauthorized access to the protected portions of the data is not to the unprotected form of the protected portions of the data.

36. A process control system comprising a device for controlling access to data, the data comprising protected data portions and rules defining access rights to the data, the device comprising:

means for storing the rules; and

an access mechanism for accessing the unprotected form of the protected data portions only in accordance with the rules, whereby output of an unprotected form of the protected data portions is permitted by the access mechanism only in such manner as is permitted by the rules.

37. A process control system as in claim **36** wherein the rules defining access rights include at least one internal rule built in the access mechanism.

38. A general purpose computer system comprising

a device for controlling access to data, the data comprising protected data portions and rules defining access rights to the data, the device comprising:

storage means for storing the rules; and

an access mechanism for accessing the unprotected form of the protected data portions only in accordance with the rules, whereby user access to an unprotected form of the protected data portions is permitted by the access mechanism only if the rules indicate that the user is allowed to access the protected portions of the data.

39. A computer system as in claim **38** wherein the rules defining access rights include at least one internal rule built in the access mechanism.

40. A computer system as in claim **38** wherein the system is implemented at various levels, and wherein at least one low level effectively defines a virtual machine in which the access mechanism is implemented, and wherein

mechanisms implemented at each level of system implementation can invoke the levels below their level of implementation.

41. A computer system as in claim **40** wherein the various levels of the computer system comprise:

an application environment level;

an operating system (OS) level which is at a lower level than the application environment level; and

a Basic Input/Output System (BIOS) level which is lower than OS level, and wherein the access mechanism is preferably implemented at or below the BIOS level.

42. A computer system as in claim **40** wherein the implementation of the access mechanism prevents a user from by-passing the access mechanism and thereby prevents a user circumventing rule enforcement by the access mechanism.

43. A computer system as in claim **40** wherein a mechanism implemented at a particular level can invoke only its implementation level and the level immediately below its implementation level.

* * * * *

EXHIBIT E

US007634666B2

(12) **United States Patent**
Cheng et al.

(10) **Patent No.: US 7,634,666 B2**
(45) **Date of Patent: Dec. 15, 2009**

(54) CRYPTO-ENGINE FOR CRYPTOGRAPHIC PROCESSING OF DATA

(75) Inventors: **Lee Ming Cheng**, Hong Kong (HK); **Ting On Ngan**, Hong Kong (HK); **Ka Wai Hau**, Hong Kong (HK)

(73) Assignee: **Cityu Research Limited**, Kowloon, Hong Kong (HK)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 509 days.

(21) Appl. No.: **10/641,869**

(22) Filed: **Aug. 15, 2003**

(65) **Prior Publication Data**

US 2005/0036617 A1 Feb. 17, 2005

(51) **Int. Cl.**
G06F 17/10 (2006.01)

(52) **U.S. Cl.** **713/191**; 713/189; 713/192; 380/30

(58) **Field of Classification Search** .... None
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,316,055	A	2/1982	Feistel	
4,484,301	A *	11/1984	Borgerding et al.	708/632
4,891,781	A *	1/1990	Omura	708/670
6,230,179	B1 *	5/2001	Dworkin et al.	708/492
6,397,241	B1 *	5/2002	Glaser et al.	708/625
6,671,709	B2 *	12/2003	Glaser et al.	708/492
7,027,597	B1 *	4/2006	Stojancic et al.	380/28
7,277,540	B1 *	10/2007	Shiba et al.	380/28

FOREIGN PATENT DOCUMENTS

WO WO 00/46954 8/2000

OTHER PUBLICATIONS

Pseudorandom Generator Based on Clipped Hopfield Neural Network; IEEE 1998; Cheng Et Al.

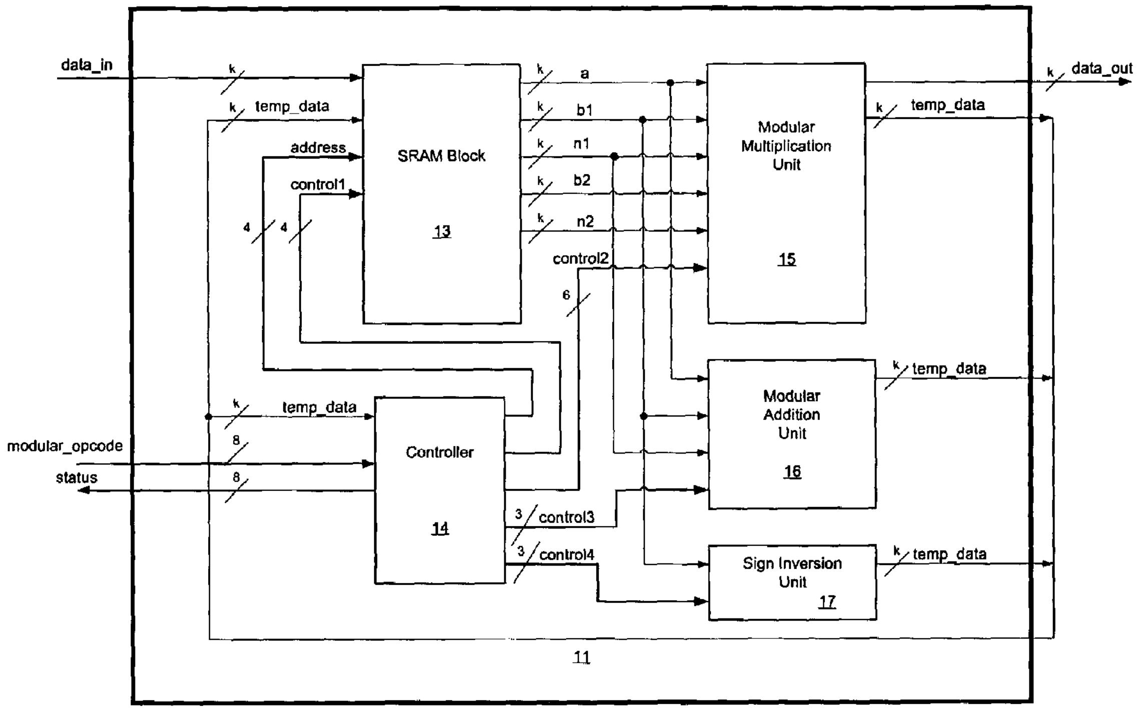
* cited by examiner

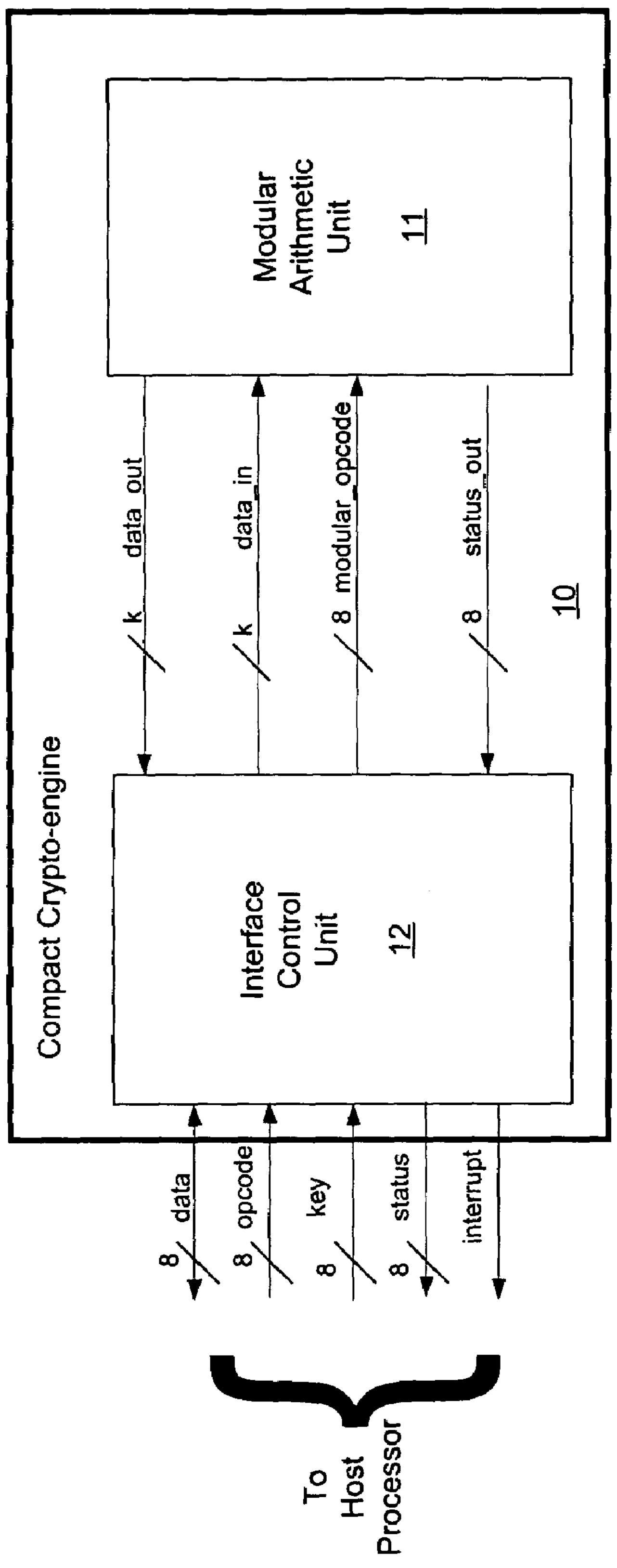
Primary Examiner—Jung Kim
(74) *Attorney, Agent, or Firm*—Alix, Yale & Ristas, LLP

(57) **ABSTRACT**

A crypto-engine for cryptographic processing has an arithmetic unit and an interface controller for managing communications between the arithmetic unit and a host processor. The arithmetic unit has a memory unit for storing and loading data and arithmetic units for performing arithmetic operations on the data. The memory and arithmetic units are controlled by an arithmetic controller.

11 Claims, 8 Drawing Sheets





Compact Crypto-engine
Modular Arithmetic Unit
11
Interface Control Unit
12
k
data_out
k
data_in
8
modular_opcode
8
status_out
10
8
data
8
opcode
8
key
8
status
interrupt
To Host Processor

FIGURE 1

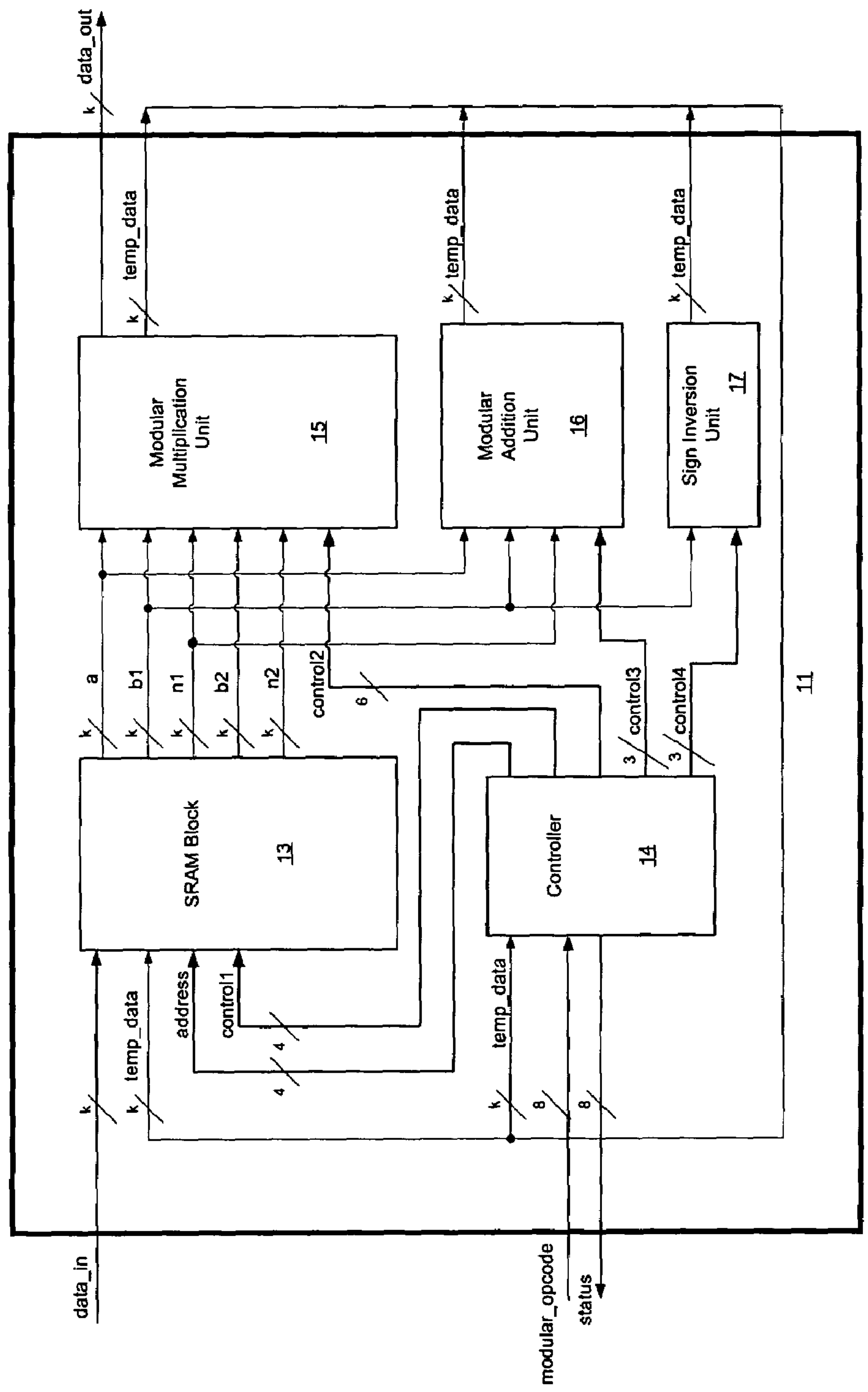

data_out
temp_data
Modular Multiplication Unit
15
Modular Addition Unit
16
Sign Inversion Unit
17
a
b1
n1
b2
n2
control2
control3
control4
6
3
k
SRAM Block
13
Controller
14
temp_data
address
control1
4
8
11
data_in
modular_opcode
status

FIGURE 2

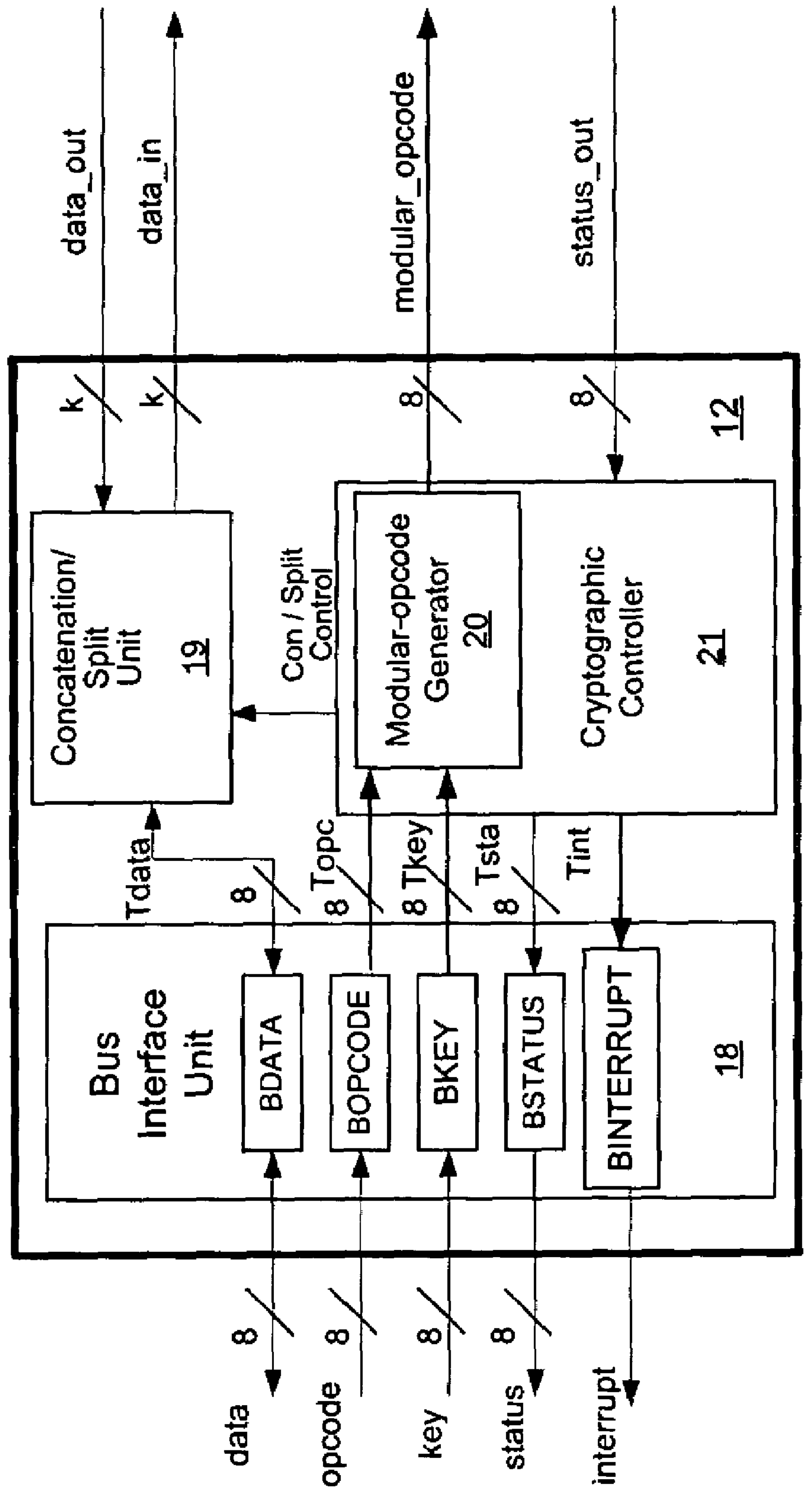

data_out
data_in
modular_opcode
status_out
k
k
8
8
12
Concatenation/ Split Unit 19
Con / Split Control
Modular-opcode Generator 20
Cryptographic Controller 21
Tdata
8
Topc
8
Tkey
8
Tsta
8
Tint
Bus Interface Unit
BDATA
BOPCODE
BKEY
BSTATUS
BINTERRUPT
18
8
8
8
8
data
opcode
key
status
interrupt

FIGURE 3

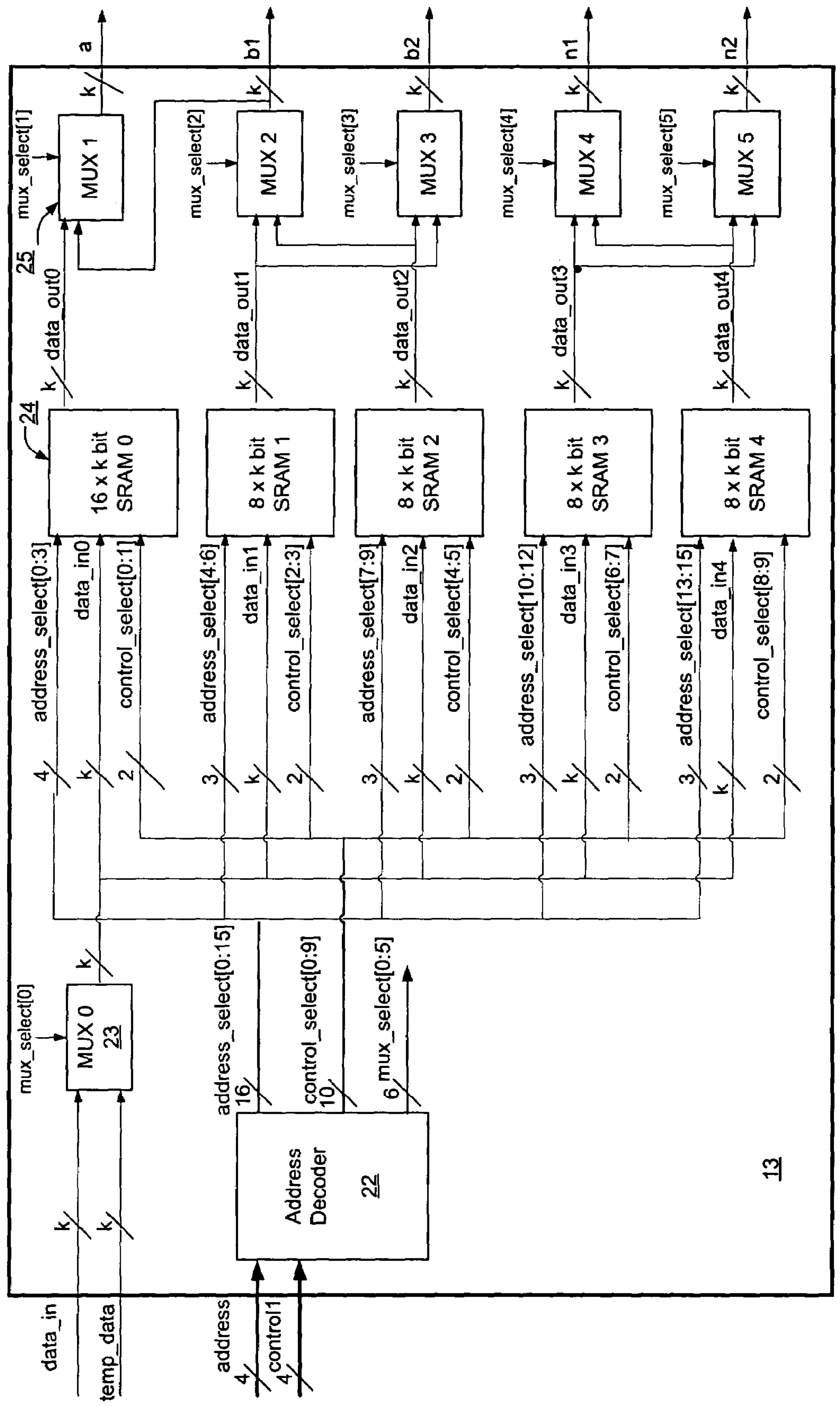
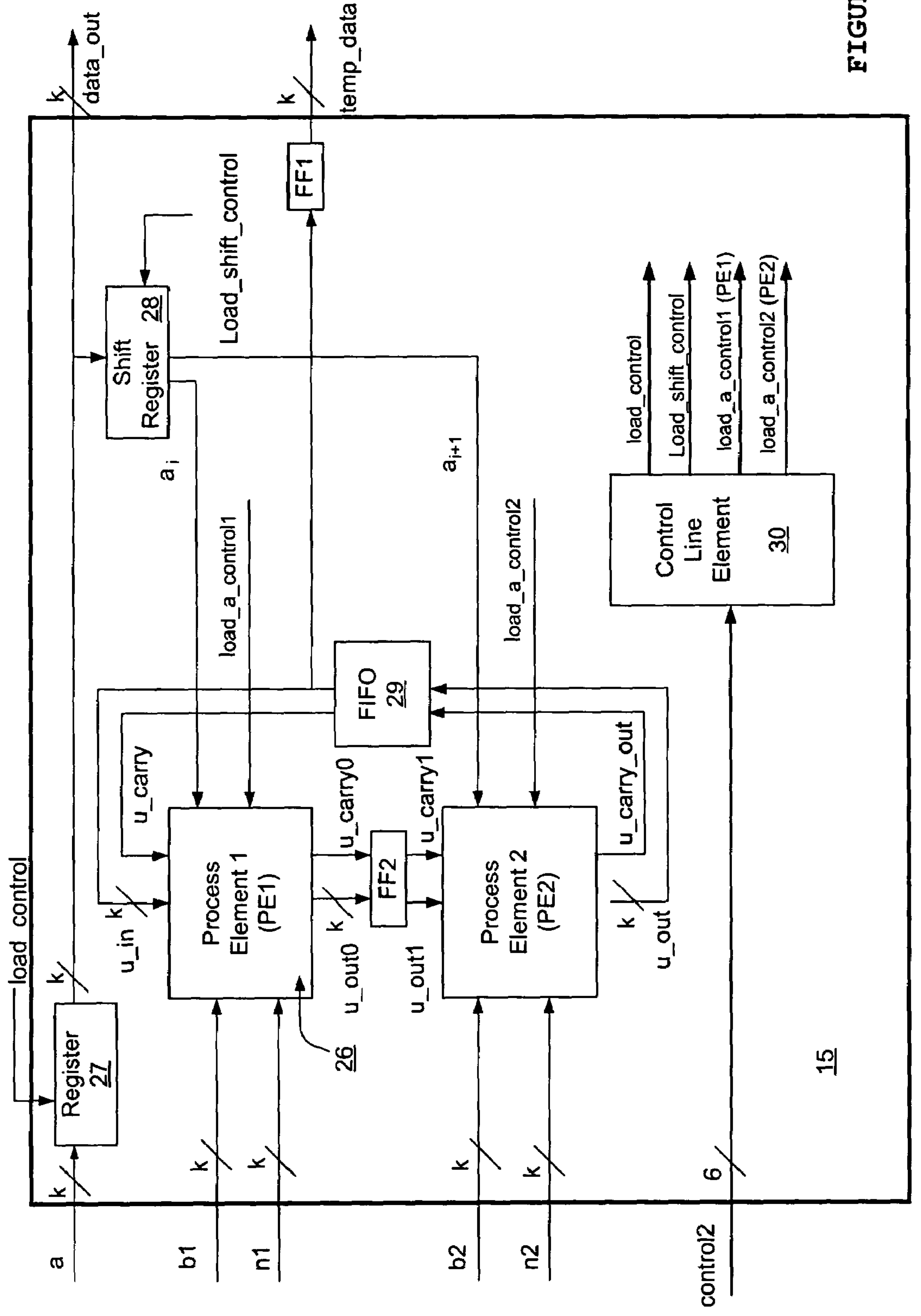
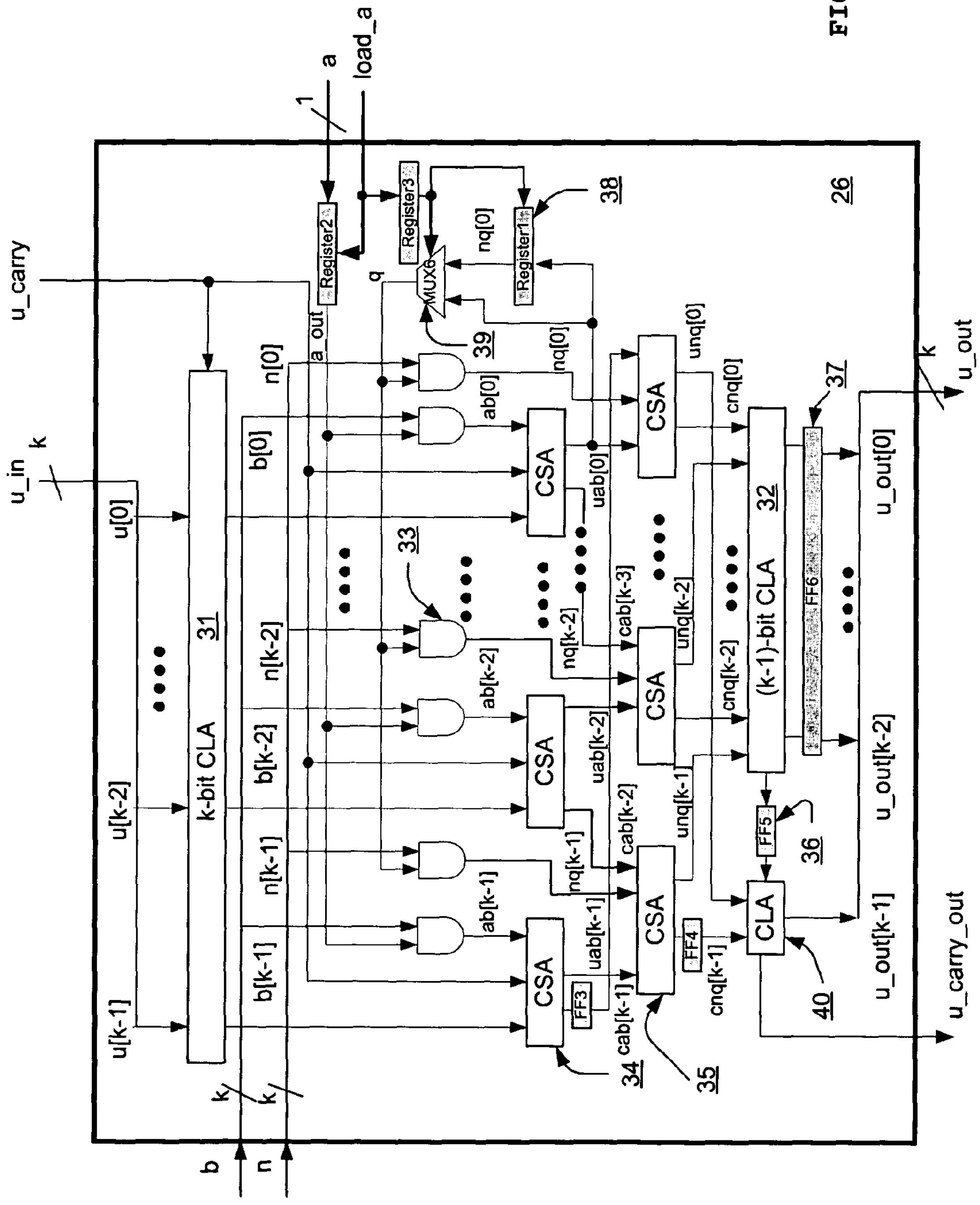

data_in
temp_data
address
control1
Address Decoder
22
address_select[0:15]
control_select[0:9]
mux_select[0:5]
MUX 0
23
mux_select[0]
16 x k bit SRAM 0
24
8 x k bit SRAM 1
8 x k bit SRAM 2
8 x k bit SRAM 3
8 x k bit SRAM 4
address_select[0:3]
data_in0
control_select[0:1]
address_select[4:6]
data_in1
control_select[2:3]
address_select[7:9]
data_in2
control_select[4:5]
address_select[10:12]
data_in3
control_select[6:7]
address_select[13:15]
data_in4
control_select[8:9]
data_out0
data_out1
data_out2
data_out3
data_out4
25
MUX 1
MUX 2
MUX 3
MUX 4
MUX 5
mux_select[1]
mux_select[2]
mux_select[3]
mux_select[4]
mux_select[5]
a
b1
b2
n1
n2
13

FIGURE 4



a
b1
n1
b2
n2
control2
k
6
Register 27
load_control
u_in
data_out
Shift Register 28
Load_shift_control
a_i
a_i+1
FF1
temp_data
load_a_control1
load_a_control2
FIFO 29
u_carry
u_carry0
u_carry1
u_carry_out
FF2
Process Element 1 (PE1)
Process Element 2 (PE2)
u_out0
u_out1
u_out
26
Control Line Element 30
load_control
Load_shift_control
load_a_control1 (PE1)
load_a_control2 (PE2)
15

FIGURE 5



u_carry
u_in
k
a
load_a
1
Register2
Register3
Register1
MUX6
q
a_out
nq[0]
38
39
26
u[0]
u[k-2]
u[k-1]
k-bit CLA
31
n[0]
b[0]
n[k-2]
b[k-2]
n[k-1]
b[k-1]
33
ab[0]
ab[k-2]
ab[k-1]
nq[k-2]
nq[k-1]
CSA
uab[0]
uab[k-2]
uab[k-1]
unq[0]
unq[k-2]
unq[k-1]
cab[k-3]
cab[k-2]
cab[k-1]
cnq[0]
cnq[k-2]
cnq[k-1]
FF3
FF4
FF5
FF6
(k-1)-bit CLA
32
37
36
CLA
40
34
35
u_out[0]
u_out[k-2]
u_out[k-1]
u_out
u_carry_out
b
n

FIGURE 6

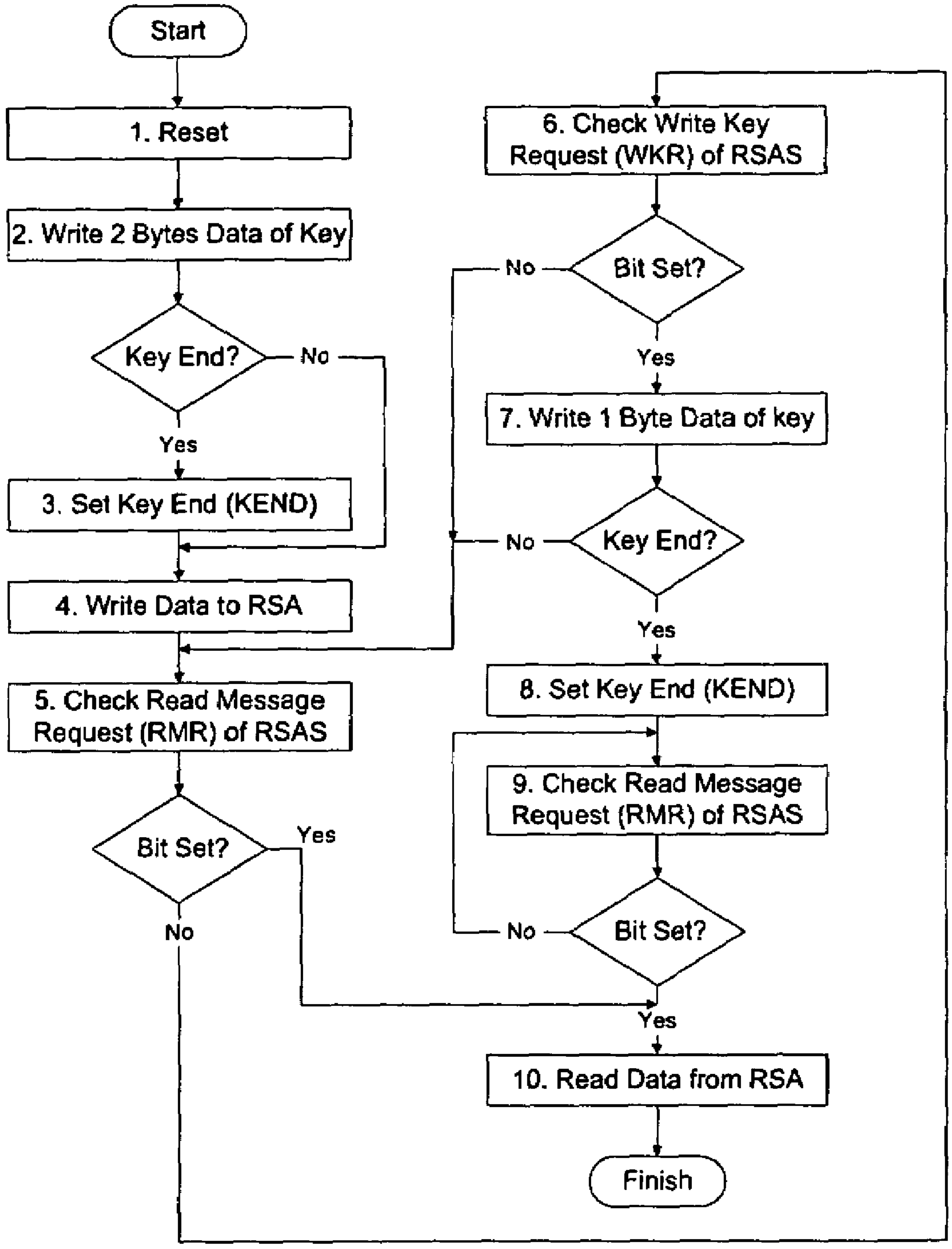

Start
1. Reset
2. Write 2 Bytes Data of Key
Key End?
No
Yes
3. Set Key End (KEND)
4. Write Data to RSA
5. Check Read Message Request (RMR) of RSAS
Bit Set?
Yes
No
6. Check Write Key Request (WKR) of RSAS
Bit Set?
No
Yes
7. Write 1 Byte Data of key
Key End?
No
Yes
8. Set Key End (KEND)
9. Check Read Message Request (RMR) of RSAS
Bit Set?
No
Yes
10. Read Data from RSA
Finish

FIGURE 7

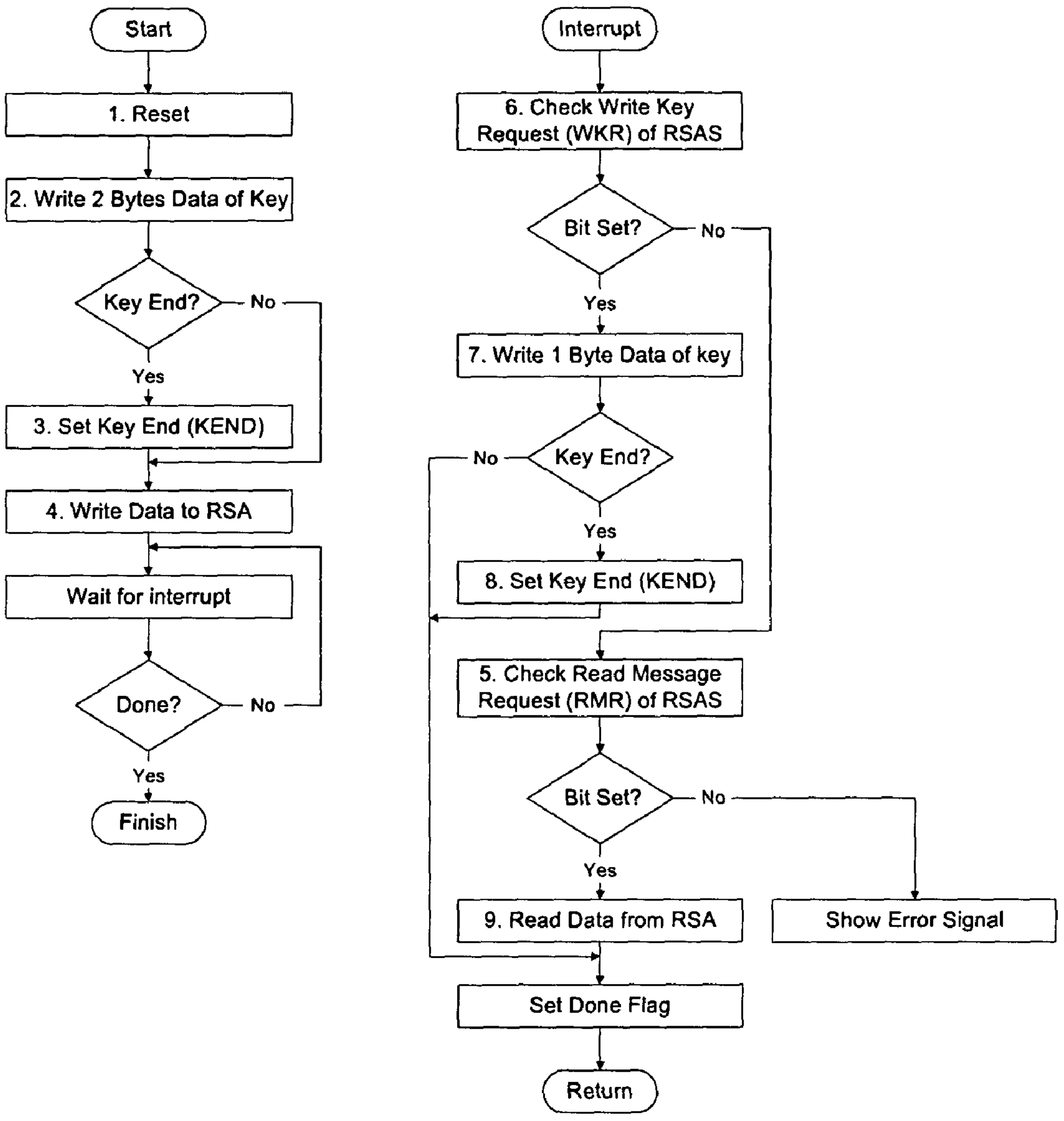

Start
1. Reset
2. Write 2 Bytes Data of Key
Key End?
No
Yes
3. Set Key End (KEND)
4. Write Data to RSA
Wait for interrupt
Done?
No
Yes
Finish
Interrupt
6. Check Write Key Request (WKR) of RSAS
Bit Set?
No
Yes
7. Write 1 Byte Data of key
Key End?
No
Yes
8. Set Key End (KEND)
5. Check Read Message Request (RMR) of RSAS
Bit Set?
No
Yes
9. Read Data from RSA
Show Error Signal
Set Done Flag
Return

FIGURE 8

CRYPTO-ENGINE FOR CRYPTOGRAPHIC PROCESSING OF DATA

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention relates to crypto-engines for cryptographic processing of data. More particularly, the invention relates to a crypto-engine capable of executing either Rivest-Shamir-Adleman (RSA) or Elliptic Curve Cryptography (ECC) public key encryption protocols.

2. Description of Prior Art

The RSA public-key cryptosystem devised by Rivest, Shamir and Adleman and the EEC cryptosystem devised by Koblitz and Miller are two common algorithms adopted by public key infrastructures.

RSA involves a computation of the exponentiation and modulo of product of two large prime numbers whereas ECC is based on computations with points on an elliptic curve. To achieve faster speed, hardware architectures are normally used to implement these algorithms.

In RSA, the main basic operation is the modular multiplication. When the ECC is implemented over the field GF(p), where p is a large prime number, the main basic operations are also modular multiplication. Thus the two algorithms share a common operation. However, in known hardware architectures resources cannot be shared by the algorithms and reused.

SUMMARY OF THE INVENTION

It is an object of the present invention to provide a hardware based crypto-engine for asymmetric cryptograhic processing using RCA or ECC algorithms. It is a further object of the invention to provide a crypto-engine that operates as a coprocessor to a host processor.

According to the invention there is provided a crypto-engine for cryptographic processing of data comprising an arithmetic unit operable as a co-processor for a host processor and an interface controller for managing communications between the arithmetic unit and host processor, the arithmetic unit including:

- a memory unit for storing and loading data,
- a multiplication unit, an addition unit and a sign inversion unit for performing arithmetic operations on said data, and
- an arithmetic controller for controlling the storing and loading of data by the memory unit and for enabling the multiplication, addition and sign inversion units.

Preferably, the memory unit comprises:

- an input switch for selecting input/interim data, a plurality of Static Random Access Memory elements for receiving and storing the input/interim data from the input switch,
- a plurality of output switches connected to the memory elements, and
- an address controller for controlling flow of the data through the switches and memory elements.

Preferably, the multiplication unit comprises:

- a register to pre-store the multiplier data,
- a pair of multiplication elements for performing multiplication,
- a shift register to load the multiplier data bitwise into the multiplication elements, and
- a first-in-first-out register for synchronizing data movement between the multiplication elements.

Preferably, the multiplication elements comprise a bitwise segmented multiplier, a bitwise segmented multiplicand, and a modulo for performing modular multiplication of the multiplier and multiplicand according to the modulo value.

Preferably, the interface controller comprises

- a bus interface for connecting high frequency manipulated data inside the arithmetic unit with the lower frequency manipulated data in the host processor,
- a concatenater/splitter for merging or splitting data width, and
- a cryptographic controller for generating status and interrupt signals for the host processor and having a op-code generator for generating the op-code signals for the arithmetic unit to select RSA or ECC operations and to synchronize the timing discrepancy of heterogeneous processing.

Further aspects of the invention will become apparent from the following description, which is given by way of example only.

BRIEF DESCRIPTION OF THE DRAWINGS

Embodiments of the invention will now be described by way of example only and with reference to the accompanying drawings in which:

FIG. **1** is a block diagram of a compact crypto-engine for asymmetric cryptographic processing according to the invention,

FIG. **2** is a block diagram of a modular arithmetic unit,

FIG. **3** is a block diagram of an interface control unit,

FIG. **4** is a block diagram of Static Random Access Memory (SRAM) Block,

FIG. **5** is a block diagram of a modular multiplication unit,

FIG. **6** is a block diagram of a processor element,

FIG. **7** is a flow diagram of RSA implementation example using polling mode, and

FIG. **8** is a flow diagram of an RSA implementation example using interrupt mode.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

In the invention a common architecture platform for the two algorithms, RSA and ECC, whose inputs are taken in two different forms, is used to manipulate the two asymmetric encryption algorithms. In the preferred embodiment the combining function is restricted to the computational engine, i.e. modular manipulation. This relies heavily on the low-bit, say 8 bit, processor software to complete the design. Thus, three design considerations must are taken into account. These considerations are:

1) hardware optimization for both RSA and ECC implementation with the best speed/resource trade off,

2) the amount of design/module reuse and hardware sharing of the two protocols, and

3) the asynchronous executing of the hardware modules in much higher speed than the processor communicating with it, i.e. heterogeneous processing.

The preferred embodiment of the present invention provides a compact crypto-engine capable of executing asymmetric cryptographic algorithms including both RSA and ECC protocols and has heterogeneous computation ability running at a higher internal clock speed.

Referring to FIG. **1**, the preferred embodiment of a compact crypto-engine **10** comprises a Modular Arithmetic Unit (MAU) **11** and an Interface Control Unit (ICU) **12**. The inputs and outputs of the ICU are provided from/to a host processor (not shown) such as a personal, network computer or Digital Signal Processor. The host processor provides an 8-bit 'data' transput (input and output) to and from ICU **12**, and 8-bit 'key' and operation code ('opcode') inputs to ICU **12**. The ICU **12** has an 8-bit 'status' and a 1-bit 'interrupt' output to signal the host processor. Communication between the ICU **12** and MAU **11** comprises a k-bit 'data_in' and a 8-bit 'modular_opcode' signals from the ICU **12** to the MAU **11**, and a k-bit 'data_out' and a 8-bit 'status_out' signals from the MAU **11** to the ICU **12**.

Referring to FIG. **2**, the MAU **11** comprises an SRAM Block **13**, a Controller **14**, a Modular Multiplication Unit (MMU) **15**, a Modular Addition Unit (MADU) **16** and a Sign Inversion Unit (SIU) **17**. The outputs k-bit 'data_in' of ICU **12**, k-bit 'temp_data' of MMU **15**/MADU **16**/SIU **17**, 4-bit 'address' and 4-bit 'control1' of Controller **14** go into SRAM Block **13**. The output k-bit 'a/b1/b2/n1/n2' of SRAM Block **13** goes to MMU **15**. The output k-bit 'a/b1/n1' of SRAM Block **13** goes to MADU **16**. The output k-bit 'b1' of SRAM Block **13** goes to SIU **17**.

The outputs 8-bit 'modular_opcode' of ICU **12** and k-bit 'temp_data' of MMU **15**/MADU **16**/SIU **17** go to Controller **14**. The outputs 4-bit 'address/control1' of Controller **14** goes to SRAM Block **13**. The output 6-bit 'control2' goes to MMU **15**. The output 3-bit 'control3' of Controller **14** goes to MADU **16**. The output 3-bit 'control4' of Controller **14** goes to SIU **17**. The 8-bit 'status_out' of Controller **14** goes to ICU **12**. The outputs k-bit 'a/b1/b2/n1/n2' of SRAM Block **13** and 6-bit 'control2' of Controller **14** go to MMU **15**. The output k-bit 'data_out' of MMU **15** goes to ICU **12** and the output k-bit 'temp_data' of MMU **15** goes to SRAM Block **13** and Controller **14**.

The outputs k-bit 'a/b1/n1' of SRAM Block **13** and 3-bit 'control3' of Controller **14** go to MADU **16**. The output k-bit 'temp_data' of MADU **16** go to SRAM Block **13** and Controller **14**. The outputs k-bit 'b1' of SRAM Block **13** and 3-bit 'control4' of Controller **14** go to SIU **17**. The output k-bit 'temp_data' of SIU **17** goes to SRAM Block **13** and Controller **14**.

Referring to FIG. **3**, the Interface Control Unit **11** comprises a Bus Interface Unit (BIU) **18**, a Concatenation/Split Unit (CSU) **19** and a Modular-opcode Generator (MOG) **20** embedded into a Cryptographic Controller (CrC) **21**. The 8-bit transput (input and output) 'data' of buffer BDATA in BIU **18** is provided to the host processor. The 8-bit outputs 'opcode' and 'key' from the host processor are provided to the buffer BOPCODE and BKEY respectively in the BIU **18**. The 8-bit output 'status' and 1-bit output 'interrupt' of BSTATUS and BINTERRUPT in BIU **18** respectively are provided to the host processor. In the preferred embodiment, the ICU provides buffers to handle heterogeneous operation and the 'interrupt' signal to synchronize the data exchange. This allows the crypto-engine **10** to operate at a different clock speed to the host processor.

The 8-bit transput 'Tdata' of Buffer BDATA in BIU **18** is provided to the Concatenation/Split Unit **19**. The 8-bit outputs 'Topc' and 'Tkey' of buffer BOPCODE and BKEY respectively in the BIU **18** are provided to the Modular-opcode Generator (MOG) **20** inside Cryptographic Controller (CrC) **21**. The outputs 8-bit 'Tsta' and 1-bit 'Tint' generated from the 'status_out' signal in the CrC **21** are provided to the BIU **18**. The k-bit output 'data_in' of Concatenation/Split Unit (CSU) **19**, generated by cascading a sequence of 8-bit 'Tdata', is provided to MAU **11**. The k-bit output 'data_out' of MAU **11**, converted to a sequence of 8-bit 'Tdata', is provided to Concatenation/Split Unit (CSU) **19**. The 8-bit output 'module_opcode' of MOG **20**, generated from signals

'Topc' and 'Tkey', is provided to MAU **11**. The 8-bit output 'status_out' of MAU **11** is provided to CrC **21** to generate the 8-bit 'Tsta' and 1-bit 'Tint' signals.

Referring to FIG. **4**, the Static Random Access Memory (SRAM) block **13** comprises an Address Decoder **22**, a plurality of switches MUX**0** **23** and MUX**1**/MUX**2**/MUX**3**/MUX**4**/MUX**5** **25**, a plurality of memory blocks **24** comprising one 16×k-bit SRAM**0** and four 8×k-bit SRAM**1**/SRAM**2**/SRAM**3**/SRAM**4**/SRAM**5**. In the preferred embodiment there are a total of 3×10^{24}-bit SRAM blocks to store the 5 parameters 'a/b1/n1/b2/n2' for 1024-bit RSA modular multiplication in various stages or to store 192-bit ECC temporary data. The gate counts required for storing of interim manipulation results are substantially reduced.

To ameliorate the overflow problems that may be encountered during the modular multiplication calculation in MMU **15**, a memory-size-expansion approach is adopted with according to the memory block size provided by Integrated Circuit fabrication supplier, say a 1152-bit memory for a 1024-bit manipulation.

Another preferred approach to overcome the overflow problem is to provide an "overflow control unit" with additional one bit for checking, say 1025-bit memory for 1024-bit manipulation.

Still referring to FIG. **4**, the 4-bit outputs 'address' and 'control1' of Controller **14** are provided to Address Decoder **22** to generate one 16-bit 'address_select[0:15]' output, one 10-bit 'control_select[0:9]' output and one 6-bit 'mux_select [0:5]' output. The output first bit 'mux_select[0]' of Address Decoder **22** is provided to switch MUX**0** **23** to select either k-bit 'data_in' outputted by ICU **12** or k-bit 'temp_data' outputted by MMU **15**/MAU **16**/SIU **17**. The outputs k-bit 'data_in 0', 'data_in1', 'data_in2', 'data_in3', and 'data_in4' of MUX**0** **23** are provided to SRAM**0**, SRAM**1**, SRAM**2**, SRAM**3** and SRAM**4** **24** respectively.

The output 3-bit address_select[0:3], address_select[4:6], address_select [7:9], address_select [10:12] and address_select[13:15] of Address Decoder **22** is provided to SRAM**0**, SRAM**1**, SRAM**2**, SRAM**3** and SRAM**4** **24** respectively. The output 2-bit control_select[0:1], control_select[2:3], control_select [4:5], control_select [6:7] and control_select[8:9] of Address Decoder **22** are provided to SRAM**0**, SRAM**1**, SRAM**2**, SRAM**3** and SRAM**4** **24** respectively.

SRAM**0**, SRAM**1**, SRAM**2**, SRAM**3** and SRAM**4** receive respective signals 'address_select[0:15]', 'data_in 0'/'data_in1'/'data_in2'/'data_in3'/'data_in4 and 'control_select[0:9]' to generate respective k-bit outputs 'data_out0', 'data_out1', 'data_out2', 'data_out3' and 'data_out4'.

The 1-bit outputs 'mux_select[1]', 'mux_select[2]', 'mux_ select[3]', 'mux_select[4]'and 'mux_select[5]' of Address Decoder **22** control switches **25** to select between MUX**1** inputs 'data_out0' or 'b1', MUX**2** and MUX**3** inputs 'data_out1' or 'data_out2' and MUX**4** and MUX**5** inputs 'data_out3' or 'data_out4'.

Referring to FIG. **2**, the k-bit outputs 'a', 'b1', 'b2', 'n1' and 'n2' of switches **25** are provided to MMU **15**; outputs 'a', 'b1' and 'n1' are provided to MAU **16**; and output 'b1' is provided to SIU **17**.

Referring to FIG. **5**, the Modular Multiplication Unit MMU **15** comprises a pair of Process Elements PE**1** **26** and PE**2** link up with a Flop-flip (FF), a Register **27**, a Shift Register **28**, a First in First Out Flip-flop (FIFO) **29** and a Control Line Element (CLE) **30**. The 6-bit output 'control2' of Controller **14** is provided to Control Line Element **30** and is decoded into a plurality of outputs 'load_control', 'load_shift_control', 'load_a_control1' (PE**1**) and 'load_a_control2' (PE**2**).

The k-bit output 'a' of SRAM Block **13** is provided to Register **27**. The k-bit output 'data_out' of Register **27** is provided to Shift Register **28** and to ICU **12** when the output 'load_control' of CLE **30** is set.

The 1-bit outputs 'a_i' and 'a_{i+1}' of Shift Register **28** are provided to Process Element **1** (PE**1**) **26** and Process Element **2** (PE**2**) respectively when the output 'load_shift_control' of CLE **30** is set.

In the preferred embodiment the interim data 'U_out' and 'u_carry_out' are included with (k+1)-bit instead of normal (2×k)-bit for logic gate size (physical hardware size) reduction and the FIFO **29** is used as a delay line for the inputs k-bit 'u_out' and 1-bit 'u_carry_out' of PE**2** to provide the inputs k-bit 'u_in' and 1-bit 'u_carry' of PE**1**. The k-bit output 'u_in' of FIFO **29** is provided to a Flip-flop (FF**1**) and the k-bit output 'temp_data' of FF**1** is provided to SRAM Block **13**.

The k-bit outputs 'b1' and 'n1' of SRAM Block **13**, the outputs k-bit 'u_in' and 1-bit 'u_carry' of FIFO **29**, the output 'a_i' of Shift Register **28** and the outputs 1-bit 'load_a_control1' (PE**1**) of CLE **30** are provided to Process Element **1** (PE**1**) to generate the outputs k-bit 'u_out0' and 1-bit 'u_carry0'. The outputs k-bit 'u_out0' and 1-bit 'u_carry0' are provided to Flip-flop (FF**2**) to generate the outputs k-bit 'u_out1' and 1-bit 'u_carry1'.

The k-bit outputs 'b2' and 'n2' of SRAM Block **13**, the outputs k-bit 'U_out1' and 1-bit 'u_carry1' of Flip-flop (FF**2**), the output 'a_{i+1}' of Shift Register **28** and the outputs 1-bit 'load_a_control2' of CLE **30** are provided to Process Element **2** (PE**2**) to generate the outputs k-bit 'u_out' and 1-bit 'u_carry_out'. The outputs k-bit 'u_out' and 1-bit 'U_carry_out' are provided to FIFO **29** to generate the outputs k-bit 'u_min' and 1-bit 'u_carry'.

Referring to FIG. **6**, the processor elements (PEs) implement Montgomery's multiplication to generate the modular multiplication. By defining

$$A=\sum_{i=0}^{m-1}a_i2^i,\quad B=\sum_{i=0}^{m-1}b_i2^i;\quad N=\sum_{i=0}^{m-1}n_i2^i\ \text{ and }\ U=\sum_{i=0}^{m-1}u_i2^i$$

as the multiplier, multiplicand, modulo and modular product (result) respectively, for m bit integers where $\{a_i, b_i, n_i, u_i\}E\{0,1\}$, the basic algorithm for Montgomery's multiplication is given as follows:

```
Module PE(A,B,U,N,m)
{$U_{-1}$ := 0;
for i = 0 to m do
    $q_i$ := ($U_{i-1}$ + $a_i$ B) mod 2; //LSB of $U_{i-1} = u_{0,i-1}$
    $U_i$ := ($U_{i-1}$ + $q_i$N + $a_i$B) div 2
endfor
return $U_m$
}
```

In order to optimize the Process Element (PE) sizes for a compact hardware implementation, instead of full m-size PE elements, k-size (where m=exk) PE pairs are included and parameters A^j, B^j, N^j and U^j are included where

$$A=\sum_{j=0}^{e-1}A^j,\quad B=\sum_{j=0}^{e-1}B^j,\quad N=\sum_{j=0}^{e-1}N^j\ \text{ and }\ U=\sum_{j=0}^{e-1}U^j.$$

The algorithm is modified into:

```
//where superscripts = blocks, subscripts = bits and for
$U_{i-1} = u_{0,i-1}$, 0 is the first outer-loop.
    Module PE(A, B, U, N, m)
    {$U_{-1}$ := 0;
    for i = 0 to m do
    // $q_i$ is implemented using MUX6 39 and CSA 34
        $q_i := {}_{0,i-1} + a_ib_0$;
        $(u\_carry, U_i^0) = a_iB^0 + U_{i-1}^0$;    // implemented using CSA 34
        $(u\_carry, U_i^0) = U_i^0 + q_iN^0 + u\_carry$;
        for j = 1 to e − 1 do
  // perform $(u\_carry, U_i^j) = a_iB^j + U_{i-1}^j + q_iN^j + u\_carry$;
  // implement using CSA 34, i.e. $U_i^j = (a_i \& B^j) \oplus U_{i-1}^j \oplus u\_carry$
  // $u\_carry = (a_i \& B^j \& u\_carry) | (U_{i-1}^j \& u\_carry) | (a_i \& B^j \& U_{i-1}^j)$
 // results store as (cab's, uab's)
            $(u\_carry, U_i^j) = a_iB^j + U_{i-1}^j + u\_carry$;
  // implement using CSA 35, i.e. $U_i^j = (q_i \& N^j) \oplus U_i^j \oplus u\_carry$
  // $u\_carry = (q_i \& N^j \& u\_carry) | (U_i^j \& u\_carry) | (q_i \& N^j \& U_i^j)$
 // results store as (cnq's, unq's)
            $(u\_carry, U_i^j) = U_i^j + q_iN^j + u\_carry$;
 // concatenate the LSB of $U_j$ to MSB of $U_{j-1}$ as carry &
  // $U_i^{j-1} := U_i^{j-1}$ div 2, implement using CLAs 32 and 49
 // results store as (u__carry__out, u__out)
            $U_i^{j-1} := (u_{0,i}^j, U_{k-1\ldots1}^{j-1})$;
    endfor
    $U_i^{(e-1)} := (u\_carry, U_{k-1\Lambda1}^{(e-1)})$
    endfor
    Return $U_m$
    }
```

In the preferred embodiment the Process Element **26** and the modified algorithm include a k-bit Carry Look-ahead Adder (CLA) **31**, a (k−1)-bit CLA **32**, a plurality of AND gates **33**, a plurality of Carry Save Adders (CSA) level **1 34** and level **2 35**, a plurality of Flip-flops **36**, a (k−1)-bit Flip-flop **37**, registers **38**, a Multiplexer MUX6 **39** and a single CLA **40**.

The outputs k-bit 'u_in' and 1-bit 'u_carry' of FIFO **29** are provided to a k-bit CLA **31** of Process Element **1** (PE**1**) **26**. For Process Element **2** (PE**2**), the outputs k-bit 'u_out1' and 1-bit 'u_carry1' are provided to a k-bit CLA **31**. The outputs k-bit 'b' (b1 or b2) of SRAM Block **13** and k-bit 'a_out' of Register**1** are provided bitwise to a plurality of two-input AND gates **33**. The outputs k-bit 'u[0:k−1]' of k-bit CLA **31**, 1-bit ‘u_carry’ of FIFO **29** and ‘ab[0:k−1]’ of AND gates **33** are provided to level 1 CSA **34** to generate a plurality of add results ‘uab[0:k−1]’ and carry ‘cab[0:k−1]’.

The outputs 1-bit ‘q’ of MUX**6** and k-bit ‘n’ (n1 or n2) of SRAM Block **13** are provided to a plurality of AND gates to generate a k-bit output ‘nq[0:k−1]’. The outputs k-bit ‘nq[0:k−1]’ of a plurality of AND gates **33**, k-bit ‘uab[0:k−1]’and k-bit ‘cab[0:k−1]’ are provided to level 2 CSA **35** bitwise to generate a plurality of add results ‘unq[0:k−1]’ and carry ‘cnq[0:k−1]’. Preferably, the output ‘cab[k−1]’ goes through a Flip-flop (FF**3**) to bit-0 (of level 2) CSA **35**.

The outputs k-bit ‘unq[0:k−1]’and ‘cnq[0:k−1]’ of a plurality of CSAs **35** are provided to a (k−1)-bit CLA **32** and 1-bit CLA **40** to generate the outputs k-bit ‘u_out’ and 1-bit ‘u_carry_out’. Preferably, the output ‘cnq(k−1)’ of CSA goes through a Flip-flop (FF**4**) to CLA **40** and the output carry of (k−1)-bit CLA **32** goes through a Flip-flop (FF**5**) **36** to CLA **40**. Preferably, the outputs of (k−1)-bit CLA **32** go through a plurality of Flip-flops (FF**6**) **37** to generate the outputs ‘u_out [0:k−2]’ of ‘u_out’.

The outputs ‘uab[0]’ of bit-0 CSA **34** and 1-bit delayed ‘uab[0]’ of Register**1** **38** are provided to MUX**6** **39** to give output ‘q’ according to condition of an output ‘load_a’ of CLE **30**. The output ‘q’ of Register**1** **38** is generated according to the outputs ‘uab[0]’ of bit-0 CSA **34** and delayed ‘load_a’ from Register**3** of CLE **30**.

The outputs 1-bit ‘load_a’ of CLE **30** and 1-bit ‘a’ of Shift Register **28** are provided to Register**2** to generate an output of 1-bit ‘a_out’.

Embodiments of the invention have been implemented using 0.35 μm semiconductor technology. A total gate count of 15K for RSA and 20K for both RSA and ECC was utilized for k=64. The benchmark testing for a 1024 (1024-bit) RSA is summarized in Table 1 as follows with an internal clock of 22 MHz.

TABLE 1

Performance of various RSA operations

Exponent	No. of ‘1’s	No. of ‘0’s	Modulus	Computation time
17 bit[1]	2	15	1024 bit	7 ms
1024 bit[2]	512	512	1024 bit	607 ms

[1]The public key e = 2^{16} + 1 = 65537 is used.
[2]Average case, 1024-bit exponent, 50% ‘1’, 50% ‘0’ in binary representation.

The benchmark device is capable of running at 100 MHz where the computational time can be reduced to 0.18 seconds for the worst case scenario.

With the heterogeneous computation ability, the process can be executed in a much higher clock rate using phase lock clock multiplier to allow faster computational and thus transaction time.

A implementation example of an RSA coprocessor is based on four special function registers (SFRs) RSAD,

RSAO, RSAS and RSAK in a host processor for controlling and monitoring the RSA coprocessor. A brief description of the SFRs now follows:

RSA DATA (RSAD)
Bit:

7	6	5	4	3	2	1	0
RSAD.7	RSAD.6	RSAD.5	RSAD.4	RSAD.3	RSAD.2	RSAD.1	RSAD.0

The bi-directional SFR is accessed via a mnemonic RSAD. Depending on the SFR RSAS, CPU and RSA coprocessor read from and write to this register. Data X, N and M are written at the beginning by software while Data M is read at the end by hardware. The RSAD is reset to 00h by a reset. There is unrestricted read/write access to this SFR.

RSA OPCODE (RSAO)
Bit:

7	6	5	4	3	2	1	0
—	—	KEND	RST	WX	WN	RWM	RW

The RSA Opcode Register with mnemonic RSAO receives instructions to configure the operation of the RSA coprocessor. This byte is set or cleared by software for the following purpose.

KEND	Key End: This bit is set to tell the coprocessor the key writing is finished.
RST	Reset: This bit is set to reset the coprocessor synchronously.
WX	Write Precomputation Constant X: When this bit and RW are set, 128 bytes of data X are written into the coprocessor. When this bit is cleared, data X will not be written.
WN	Write Modulus N: When this bit and RW are set, 128 bytes of data N are written into the coprocessor. When this bit is cleared, data N will not be written.
RWM	Read Write Message M: When this bit and RW are set, 128 bytes of data M are written into the coprocessor. When this bit is set while RW is cleared, 128 bytes of data M are read from the coprocessor. When this bit is cleared, data M will not be read or written.
RW	Read Write Control: When this bit is set, data X, N, M will be written depends on bits WX, WN, RWM. When cleared, 128 bytes of data M are read from the coprecessor if RWM is set.

All possible combination of read/write operation:
WN
RWM
RW
Read/Write Operation
1
0
0
1
Write data X
0
1

0
1
Write data N
0
0
1
1
Write data M
1
1
0
1
Write data X and N
1
0
1
1
Write data X and M
0
1
1
1
Write data N and M
1
1
1
1
Write data X, N and M
X
X
1
0
Read data M
X
X
0
0
No operation
0
0
0
X
No operation

The RSAO is reset to 00h by a reset. There is unrestricted read/write access to this SFR.

RSA STATUS (RSAS)
Bit:

7	6	5	4	3	2	1	0
—	—	—	—	WKR	—	RMR	—

The status with mnemonic RSAS of the RSA coprocessor is expected to shown in the RSA Status Register. This byte is set or clear by hardware for the following purpose.

WKR	Write Key Request: This bit is set to request the CPU to write the next byte of key to the SFR RSAK.
RMR	Read Message Request: This bit is set to tell the CPU that the RSA operation is finish and it is ready to read the data M. It also requests the CPU to write instruction to read data M from RSAD.

The RSAS is reset to 00h by a reset.

There is restricted read only access to this SFR.

RSA KEY (RSAK)
Bit:

7	6	5	4	3	2	1	0
RSAK.7	RSAK.6	RSAK.5	RSAK.4	RSAK.3	RSAK.2	RSAK.1	RSAK.0

The SFR with mnemonic RSAK will be used to store the key. One byte of RSA key, i.e. the exponent e or d is written into this register by software, while the bit WKR of the SFR RSAS is set. The RSAK is reset to 00h by a reset. There is unrestricted read/write access to this SFR.

The procedure of control the RSA coprocessor to carry out a RSA operation is summarized in FIGS. **7** and **8**. The sequence of operation is as follows:

1. The coprocessor must be reset at the beginning of RSA operation; the Reset (RST) bit is set (RSAO=10h) and cleared (RSAO=00h) to reset the coprocessor.

2. Two bytes of RSA key are then written to RSAK, starting from the most significant byte.

3. If the key ends, i.e. the key is less than or equal to 2 bytes, set the bit KEND of RSAO (RSAO=20h) to inform the coprocessor.

4. Set the Write operation by setting appropriate bits in RSAO, followed by writing the data block(s) in the order of data X, N and M into RSAD, starting from the least significant byte of first data block. For example, if RSAO=0Fh, 3×128 bytes of data X, N, and M are written to RSAD sequentially, starting from the least significant byte of data X; If RSAO=0Bh, 2×128 bytes of data X and M are written to RSAD sequentially, starting from the least significant byte of data X; If RSAO=09h, only 128 bytes of data X is written to RSAD, starting from the least significant byte of data X.

5. Check the WKR of RSAS to see whether the RSA coprocessor request next byte of key.

6. If the WKR is set, write one byte of key to RSAK.

7. If the key ends, i.e. all bytes of key is written into RSAK, set the bit KEND of RSAO (RSAO=20h) to inform the coprocessor.

8. Check the RMR to see whether the result data is ready to be read.

9. When it is ready to read the data, the read data M instruction is assigned to the RSAO (RSAO=02h). 128 bytes of data M are read from RSAD, starting from the least significant byte of data M.

Where in the foregoing description reference has been made to methods or elements have known equivalents then such are included as if individually set forth herein.

Embodiments of the invention have been described, however it is understood that variations, improvement or modifications can take place without departure from the spirit of the invention or scope of the appended claims.

What is claimed is:

1. A crypto-engine for cryptographic processing of data comprising an arithmetic unit operable as a co-processor for a host processor and an interface controller for managing communications between the arithmetic unit and host processor, the arithmetic unit including:

a memory unit for storing and loading data, the memory unit including

an input switch for selecting input-interim data;

a plurality of Static Random Access Memory elements for receiving and storing the input/interim data from the input switch;

a plurality of output switches connected to the memory elements; and

an address controller for controlling flow of the data through the switches and memory elements

a multiplication unit, an addition unit and a sign inversion unit for performing arithmetic operations on said data, the multiplication unit, the addition unit and the sign inversion unit each having an output; and

an arithmetic controller for controlling the storing and loading of data by the memory unit and for enabling the multiplication, addition and sign inversion units;

wherein the outputs of the multiplication unit, the addition unit and the sign inversion unit are feedback to the arithmetic controller.

2. The crypto-engine of claim **1** wherein the multiplication unit comprises:

a register to pre-store the multiplier data;

a pair of multiplication elements for performing multiplication;

a shift register to load the multiplier data bitwise into the multiplication elements; and

a first-in-first-out register for synchronizing data movement between the multiplication elements.

3. The crypto-engine of claim **2** wherein the multiplication elements comprise a bitwise segmented multiplier, a bitwise segmented multiplicand, and a modulo for performing modular multiplication of the multiplier and multiplicand according to the modulo value.

4. A crypto-engine for cryptographic processing of data comprising an arithmetic unit operable as a co-processor for a host processor and an interface controller for managing communications between the arithmetic unit and host processor,

the arithmetic unit including:

a memory unit for storing and loading data;

a multiplication unit, an addition unit and a sign inversion unit for performing arithmetic operations on said data, the multiplication unit, addition unit and sign inversion unit each having an output; and

an arithmetic controller for controlling the storing and loading of data by the memory unit and for enabling the multiplication, addition and sign inversion units, wherein the outputs of the multiplication unit, an addition unit and a sign inversion unit are feedback to the arithmetic controller;

the interface controller including:

a bus interface for connecting high frequency manipulated data inside the arithmetic unit with the lower frequency manipulated data in the host processor;

a concatenater/splitter for merging or splitting data width, and

a cryptographic controller generating status and interrupt signals for the host processor and generating an op-code signal for the arithmetic unit, the arithmetic unit selecting RSA or EGO modes of operation based on the op-code signal.

5. The crypto-engine of claim **4** wherein the multiplication unit comprises:

a register to pre-store the multiplier data;

a pair of multiplication elements for performing multiplication;

a shift registerto load the multiplier data bitwise into the multiplication elements; and

a first-in-first-out register for synchronizing data movement between the multiplication elements.

6. The crypto-engine of claim **5** wherein the multiplication elements comprise a bitwise segmented multiplier, a bitwise segmented multiplicand, and a modulo for performing modular multiplication of the multiplier and multiplicand according to the modulo value.

7. The crypto-engine of claim **5** wherein the memory unit has a size substantially equal to 384 bytes and the sign inversion unit has a k-size substantially equal to 64 bits.

8. The crypto-engine of claim **1** wherein the outputs of the multiplication unit, the addition unit and the sign inversion unit are feedback to the arithmetic controller and the memory unit.

9. The crypto-engine of claim **4** wherein the memory unit has a size substantially equal to 384 bytes and the sign inversion unit has a k-size substantially equal to 64 bits.

10. The crypto-engine of claim **4** wherein the multiplication unit, the addition unit and the sign inversion unit each having an output that is feedback to the arithmetic controller.

11. The crypto-engine of claim **10** wherein the outputs of the multiplication unit, the addition unit and the sign inversion unit are feedback to the arithmetic controller and the memory unit.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO.	: 7,634,666 B2	Page 1 of 1
APPLICATION NO.	: 10/641869	
DATED	: December 15, 2009	
INVENTOR(S)	: Cheng et al.	

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

<u>Column 12</u>:
Line 18, delete “EGO” and substitute --ECC--.

Line 25, delete “registerto” and substitute --register to--.

Line 34, delete “claim 5” and substitute --claim 1--.

Signed and Sealed this

Thirtieth Day of March, 2010

David J. Kappos

David J. Kappos
Director of the United States Patent and Trademark Office

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO.	: 7,634,666 B2	Page 1 of 1
APPLICATION NO.	: 10/641869	
DATED	: December 15, 2009	
INVENTOR(S)	: Cheng et al.	

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On the Title Page:

The first or sole Notice should read --

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 983 days.

Signed and Sealed this

Ninth Day of November, 2010

David J. Kappos

David J. Kappos
Director of the United States Patent and Trademark Office